Universidad de las Ciencias Informáticas

Facultad #3

Título: Diseño del Subsistema GCPA e Implementación de los Casos de Uso críticos, del Proyecto Sistema de Gestión Fiscal.

Trabajo de Diploma para optar por el título de
Ingeniero Informático

Autor(es): Alexy Rodríguez Tamayo
Tutor(es): Ing. Yusmary Galbán Izquierdo

7 DE MAYO DEL 2009.

"...aquí está una de las tareas de la juventud: empujar, dirigir con el ejemplo la producción del hombre de mañana. Y en esta producción, en esta dirección, está comprendida la producción de si mismos..."

CHÉ

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Dirección de la Universidad de las Ciencias Informáticas de hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los 7 días del mes de mayo del año 2009.

Alexy Rodríguez Tamayo

_____

Ing. Yusmary Galbán Izquierdo

_____

DATOS DE CONTACTO

Autor: Alexy Rodríguez Tamayo.
Correo Electrónico: artamayo@estudiantes.uci.cu

Tutor: Yusmary Galbán Izquierdo.
Correo Electrónico: ygalban@uci.cu

AGRADECIMIENTOS

- Primeramente a mis padres por ser la razón de mis esfuerzos, por haberme apoyado en mis momentos tristes que pudieron llevarme al fracaso.
- A mi hermano mayor Sergio por confiar en mí y por servirme de punto de referencia en muchas de mis decisiones.
- A mi hermano Michel, su novia y a mi sobrino Kevin.
- A prima Maricela, a mi tía Zoila y a mi primo Carlos Manuel por haber estado cuando más necesitaba de ayuda, y por ser a mi juicio mi mano derecha para alcanzar este resultado.
- A mi primo Noel, mi tía Rosa Elba y a Julio por haberme ayudado a despejar en mis momentos malos.
- A mis padrinos Felo y Milagro por sus consejos y su apoyo en todo momento.
- Agradecimiento especial para mis abuelos por haber rezado por mi en mis peores momentos y así contribuyeron a que pudiera salir del problema.
- A mis tíos por estar pendiente de mis resultados y darme aliento años tras año para que alcanzara el objetivo de estos 5 años.
- A mi primo Sergio y su esposa Any por ayudarme en estos 5 años a terminar con éxito.
- A mi prima Yuleny y a Gabriel por estar pendiente de mis resultados.
- A mis amigos Santo, Payo y Pellón, por estar pendientes de mis resultados y darme aliento para llegar al final.
- A Daniel Mariano compañero de mi hermano por su ayuda en el primer año de mi carrera, que sin ella otros gallos hubieran cantado.
- A mi compañero de estudio Rayner Julio Rodríguez por su gran ayuda en los 2 primeros años de la carrera.
- A la profesora Dariela por sus consejos y ayuda en estos 5 años.
- A todas mis amistades de la universidad que fueron punto clave en este resultado.
- A mis suegros por los consejos y la ayuda que me dieron en estos 5 años.
- A mi novia Diana por haberme acompañado durante estos 5 años y apoyarme en mis momentos depresivos, además por darme el mejor regalo que puede recibir cualquier persona, un lindo niño el cual a pesar de llegar en el último año de mi carrera, ha pasado a ser el principal motivo de esforzarme por alcanzar buenos resultados.
- A mi tutora Yusmary Galbán por la ayuda prestada en la confección de este trabajo y por estar pendiente de que se terminara en tiempo.
- A Fidel y la Revolución cubana por dejarme ser parte de la UCI.

DEDICATORIA

El resultado de este trabajo es para mi mamá y mi papá que son los tutores principales de mi vida, tiene una dedicación especial para mi hermano Michel por estar siempre y ayudarme en todo lo que me hizo falta, va dirigido a mi novia por haber confiado en mí, está dedicado a mi sobrinito Kevin y a mi niño Alexy ya que serán los mas beneficiados con este resultado, es también para mis padrinos Felo y Milagro pues fue una promesa realizada el día que entre a la universidad, es un regalo para mis abuelitos que siempre los tengo presente, y fue alcanzado gracias a las personas que fueron el remedio para mi enfermedad en estos 5 años que estuve muy lejos de casa, pues ellos ocuparon esta función brindándome mucho apoyo y confianza, mi prima Maricela, mi tía Zoila y mi primo Carlos Manuel.

RESUMEN

Con la idea de mejorar la automatización de los procesos en la Fiscalía General de la República de Cuba (SGF) a fin de garantizar, con mayor economía, profesionalidad y celeridad, el control y la preservación de la legalidad, sobre la base de la vigilancia del estricto cumplimiento de la Constitución, las leyes y demás disposiciones legales, por los organismos del Estado, entidades económicas, sociales y por los ciudadanos , surge la realización de un sistema informático que gestione los procesos de la Fiscalía General de la República (SGF) en todo el país.

La esencia de este trabajo es el diseño y la implementación del módulo de Gestión de Cuadros y Personal de Apoyo (GCPA), para la realización del mismo se utilizará el Visual Paradigm para realizar el modelado de todo el ciclo de vida del proyecto, específicamente el diseño y para la implementación del sistema se trabajará con PHP5, el sistema se desarrollará sobre el Framework de Symfony en Linux (Debian) y como gestor de base de datos PostgreSQL, para así lograr la automatización con herramientas mayormente libres y multiplataformas.

El diseño y la implementación de este sistema tendrán un gran impacto social ya que se mejorará la eficiencia y la calidad de los procesos fiscales en todo el país lo cual contribuirá a mejorar la atención al cliente en estos centros de trabajo, además estandarizará el trabajo en todas las Fiscalías de la nación dotando a los jueces y fiscales de un sistema robusto y moderno que les permitirá realizar de forma segura todo el control necesario en esta esfera de trabajo.

TABLA DE CONTENIDOS

TABLA DE CONTENIDOS

INTRODUCCIÓN

Introducción

Hoy en día las diferentes sociedades del mundo se encuentran regidas por normas y procedimientos jurídicos a los que se les denomina como ordenamiento jurídico ó Sistemas de Derecho, estos para su creación van aparejados de varios factores sociales como las costumbres, tradiciones, normas, sentencias e intereses de las diferentes clases dominantes. Existen países principalmente en Iberoamérica que tiene sistemas de derechos similares al de Cuba, con peculiaridades y estructuras propias. Otros como Estados asiáticos, islámicos, los de la antigua Unión Soviética, Inglaterra y sus colonias, Estados Unidos, Puerto Rico el sistema es totalmente diferente.

En los grandes países desarrollados existen aplicaciones informáticas que ayudan al trabajo con estos engorrosos procesos, pero estas dejan siempre detalles procesales y las posibles decisiones a adoptar en manos de la experiencia de los juristas ya que esta es el arma principal en el capitalismo.

El desarrollo científico técnico alcanzado a nivel mundial en el campo de las Tecnologías Informáticas y las Comunicaciones es realmente impresionante, la mayoría de las empresas están en proceso de automatización y otra gran parte se encuentran automatizadas en vista de generar mayor calidad en sus procesos.

Actualmente en Cuba la Fiscalía General de la República se encuentra enfrascada en la informatización de todos los procesos fiscales a los que está vinculada, algunos de estos se trabajan actualmente con un software que no satisface por completo el trabajo que los fiscales realizan, debido a que la demanda de información es cada vez mayor, por lo que la rapidez y la eficiencia no es factible para continuar el proceso, la demanda de almacenamiento aumenta cada día y la estandarización del trabajo en todo el país se hace necesaria.

Uno de los procesos a automatizar en la Fiscalía es el referente a la Gestión de Cuadros y Personal de Apoyo (GCPA), en el cual el trabajo es cada día más engorroso porque muchos de los documentos generados tienen que hacerse por escrito y tienen que almacenarse muchos en copia dura ya que no existe un sistema digital que genere y almacene toda la información que los fiscales necesitan, el control de los cuadros dentro de la Fiscalía y la gestión del personal de apoyo que están vinculados a esta, como los estudiantes que cursan la carrera de derecho y están haciendo prácticas en la Fiscalía se dificulta cada día tras día. La capacitación y el control de la seguridad de acceso tanto de los cuadros como del personal de apoyo son temas que cada año que transcurre se hace más difícil de controlar, debido a que cada año aumenta la cantidad de personal en la Fiscalía General por lo que se hace necesario darle una respuesta inmediata a tan compleja situación.

Por estas razones surge la necesidad de crear un proyecto productivo que realice un sistema informático confiable y seguro, que ayude a resolver el problema en cuestión que consiste en**:**

¿Cómo facilitar el proceso de Gestión de Cuadro y Personal de Apoyo (GCPA) en el Proyecto Sistema de Gestión Fiscal, de manera que se logre un sistema robusto y flexible?

Como **objeto de estudio** de este trabajo se define el Proceso de Desarrollo de Software.

Derivándose como **campo de acción** el diseño del Subsistema Gestión de Cuadro y Personal de Apoyo (GCPA) e Implementación de los casos de uso críticos en el Proyecto Sistema de Gestión Fiscal.

La **hipótesis** de este trabajo parte de la idea, que si se realiza un buen diseño e implementación del Subsistema Gestión de Cuadro y Personal de Apoyo (GCPA), en el Proyecto Sistema de Gestión Fiscal, se logrará alcanzar un producto robusto y flexible que garantice mayor calidad en esta esfera de trabajo.

El **objetivo general** de este trabajo es: realizar el diseño y la implementación del Subsistema Gestión de Cuadro y Personal de Apoyo del Proyecto Sistema de Gestión Fiscal.

Para satisfacer este objetivo y resolver el problema planteado se proponen las siguientes **tareas**:

- Realizar un estudio de las herramientas de gestión de cuadros a nivel nacional e internacional.
- Realizar un estudio de los patrones de diseño y lenguajes de programación.
- Seleccionar los patrones de diseño y lenguajes de programación adecuados en cada caso para la elaboración del producto de software.
- Realizar un estudio de las herramientas de implementación.
- Seleccionar las herramientas de implementación que se utilizarán en el sistema y fundamentar la elección.
- Diseñar el Subsistema de Gestión de Cuadro y Personal de Apoyo (GCPA).
- Implementar los casos de uso críticos del Subsistema de Gestión de Cuadro y Personal de Apoyo (GCPA).
- Realizar un estudio de las métricas de diseño que existen en el desarrollo de software.
- Aplicar de las métricas de diseño adecuadas al producto de software implementado.

Con el cumplimiento de estas tareas se pretende entregar el diseño y la implementación de un software que esté a la altura de las exigencias actuales de la producción de software de nuestra sociedad, acorde con los estándares internacionales de catalogación, los estándares de diseño e implementación y que cumpla con todas las exigencias realizadas por los clientes.
Este sistema proveerá al personal de la Fiscalía de un gran ahorro de tiempo y afectará de forma positiva la toma de decisiones en los procesos, contribuirá de forma eficiente al control de la capacitación y acceso de todos sus miembros a la Fiscalía General, mejorando la productividad y ayudando a la construcción de la cultura socialista de nuestro pueblo, pues se logrará cumplir con las leyes de forma segura.

Métodos Científicos de investigación utilizados:
Los métodos científicos de investigación son la forma de estudiar la realidad natural, el pensamiento, los distintos fenómenos sociales con el objetivo de descubrir su esencia y sus relaciones.
Para realizar la investigación se utilizaron los siguientes métodos teóricos:

Inductivo-Deductivo
Hizo posible que con el estudio de la vieja aplicación y del trabajo en la Fiscalía, se realizara la creación de una aplicación que esté compuesta por los viejos resultados e incluya los nuevos procedimientos que necesita la antigua aplicación.

Histórico-Lógico
Ayudó a realizar un estudio histórico sobre las aplicaciones que realizaban trabajos similares, para así poder tener un mayor conocimiento y saber aplicarlo en la solución del problema presentado.

Hipotético-Deductivo
Brindó la posibilidad de determinar los principales elementos que facilitarían la modelación de los diferentes diagramas mediante las herramientas de modelado seleccionadas para realizar este trabajo.

Métodos empíricos:
Observación
Este método permitió definir que parte de la aplicación y del trabajo de la Fiscalía se debían observar, seguir constantemente la modelación del trabajo y controlar los cambios que surgen a lo largo del desarrollo de la aplicación. Además de precisar los hechos reales, que existen sobre la modelación de la aplicación y no aquellos que son ficticios o simplemente no tienen una fundamentación válida sobre la misma.

Este documento está estructurado de la siguiente manera:

Capítulo 1. Fundamentación teórica

El Capítulo 1 trata diferentes temas que ayudarán a resolver el problema en cuestión como son conocer el estado del arte de las versiones anteriores al sistema Gestión de Cuadro y Personal de Apoyo, se hará un breve análisis de las tendencias de los lenguajes de programación y se tratarán temas como métricas, Patrones de Diseño, herramientas y las tecnologías actuales utilizadas en el proceso de desarrollo de software y al finalizar se concluye el capítulo.

Capítulo 2: Arquitectura, Diseño e Implementación

El Capítulo 2 aborda temas relacionados con la arquitectura de sistema, el estilo arquitectónico seleccionado a partir del análisis realizados en el capítulo anterior, los estándares de codificación, los patrones de diseño aplicados, los componentes usados, los diferentes diagramas de clases del diseño y de iteración, finalmente se realiza la conclusión del capítulo.

Capítulo 3: Análisis de Resultados

En el Capítulo 3 se realiza un análisis de los resultados de este trabajo, se tratan las métricas aplicadas al diseño realizado y se muestran los resultados de las pruebas de caja negra realizadas al sistema para verificar que esta en condiciones de ser usado, por último se realiza la conclusión del capítulo.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se hará un estudio de los diferentes factores que ayudarán a desarrollar el software propuesto, como realizar el estado del arte de las diferentes herramientas que existen en mundo y en cuba que sean similares a este proceso, estudiar el Lenguaje de Modelado, las Metodologías de Desarrollo, los Lenguajes de Programación, las Herramientas Case, los Patrones de Diseño y estilos arquitectónicos, los Frameworks de Desarrollo así como conocer sobre las diferentes Métricas de Diseño utilizadas a lo largo del Proceso de Desarrollo. Este capítulo tendrá como objetivo principal la selección de la herramienta, la metodología, el lenguaje de programación y el Framework de Desarrollo que se utilizará en la solución propuesta.

1.2 Estado del arte de herramientas similares existentes.

En el mundo existen herramientas que trabajan con asuntos legales y ayudan al control y al cumplimiento de las leyes, algunas de estas son:

Gedex

Realiza el seguimiento completo de los expedientes jurídicos de despachos, bufetes o departamentos jurídicos. Ayuda a incrementar los beneficios, así como a ahorrar en costes de gestión. Beneficiarse (en tiempo y dinero) al informatizar los expedientes. Este es implementado en España e Hispanoamérica, en bufetes, despachos individuales y departamentos jurídicos de empresa. [3]

Compilación Jurídica

Es el software de Informática Jurídica más rápido, completo y único de Venezuela. Este software le permite en breves segundos localizar cualquier jurisprudencia, artículo de ley, comentario o información de la base de datos e interactuar con las opciones de consulta simultánea, impresión, consulta de diccionario, e imprimir modelos de documentos. [4]

ABOGest

Es una aplicación diseñada para controlar quienes y de que forma se relacionan con su bufete, que asuntos tiene en trámite y los que han llegado a su término. Es el único software en España que ha obtenido una certificación oficial de un colegio de abogados, y ha sido diseñado, pensado y desarrollado por y para los abogados. [2]

En Cuba se desarrolló en el seno de la Sociedad Cubana de Derecho e Informática un software denominado **SOFTELX**, el cual trata algunas normativas y tiene un módulo de seguimiento de asuntos, hasta cierto punto configurable por el usuario, que recoge una serie de datos útiles de los procesos. En la Fiscalía General existe una aplicación de escritorio construida en Delphi que contiene una base de datos en Access pero no recoge todos los procesos que la Fiscalía realiza, esta muy antigua y no satisface la demanda de información que generan los fiscales en su trabajo, como tampoco realiza el control del personal que trabaja en la institución.

1.3 Desarrollo de Software. Tendencias y Herramientas

Actualmente el Proceso de Desarrollo de Software bien estructurado y siguiendo correctamente los elementos que rigen al mismo tiene tendencia a mejorar cada día mas el producto final elaborado y propiciar comodidad al trabajo que realizan los desarrolladores, a continuación se especifican varios de los factores que intervienen en un buen resultado a lo largo de todo el ciclo de vida del proyecto, estos elementos ligados hacen que cada día aumente la calidad en todo el proceso de desarrollo.

1.3.1 Lenguaje Unificado de Modelado (UML).

UML es un lenguaje de modelado que contiene reglas que se combinan con elementos gráficos para conformar diagramas. UML no es una guía para realizar el análisis y diseño orientado a objetos sino que es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos.

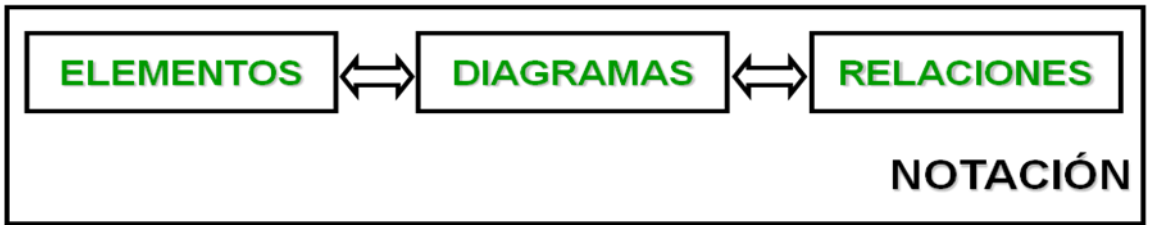

Figura: 1.3.1.1 Notación UML

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado para sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software, se utiliza para definir un sistema de software y detallar los artefactos en el sistema.
En otras palabras, es el lenguaje en el que está descrito el modelo de un sistema. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software, pero no especifica en sí mismo que metodología o proceso usar.
Es importante resaltar que UML es un "lenguaje" para especificar y no para describir métodos o procesos [5].

1.3.2 Metodologías de Desarrollo.

En el Proceso de Desarrollo de Software es importante el uso de una metodología de desarrollo ya que estas son las que definen “Quién” va hacer “Qué”, “Cuándo” y “Como” va hacerlo, lo cual ayuda a organizar el trabajo y orientar a todos los trabajadores. Partiendo de que una metodología es un proceso y de que no existe en la actualidad una metodología universal para

realizar el Proceso de Desarrollo porque estas se dividen en dos grupos: los procesos tradicionales y los ágiles con marcadas diferencias, a continuación este epígrafe abordará una breve descripción de Rational Unified Process (RUP) y Extreme Programming (XP) metodologías utilizadas en el mundo para realizar el Proceso de Desarrollo de Software.

1.3.2.1 Programación Extrema (Extreme Programming, XP).

XP es un conjunto de técnicas y prácticas para el desarrollo de software. Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Sus principios básicos son dos: la mejora de la comunicación con los usuarios, para retroalimentar el proceso de desarrollo; y obtener cuanto antes un programa que haga algo, es decir un producto tangible que partiendo de este, se puedan ir retroalimentando y en la rectificación de errores se puedan ir agregando nuevas características hasta lograr el resultado deseado. Esta es una de las características importantes de XP, la retroalimentación continua entre el cliente y el equipo de desarrollo, la comunicación fluida entre todos los participantes y la simplicidad en las soluciones implementadas, además de enfrentar con gran valor los cambios presentados ya que estos no dependen de la fase o etapa. La constante comunicación con el usuario o cliente es otra característica muy importante a tener en cuenta con el uso de esta metodología, ya que el usuario o cliente pasa a formar parte del equipo de trabajo.
XP es adecuado para realizar proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. Estas técnicas se le aplican a proyectos con un equipo de desarrollo medio grande, para solucionar un problema no trivial.

El ciclo de desarrollo consiste (a grandes rasgos) en los siguientes pasos:

1. El cliente define el valor de negocio a implementar.
2. El programador estima el esfuerzo necesario para su implementación.
3. El cliente selecciona que construir, de acuerdo con sus prioridades y las restricciones de tiempo.
4. El programador construye ese valor de negocio.
5. Se regresa al paso 1.

Los roles de acuerdo con la propuesta original de Beck son: [30]

1. **Programador**. El programador escribe las pruebas unitarias y produce el código del sistema.
2. **Cliente**. Escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuales se implementan en cada iteración centrándose en aportar mayor valor al negocio.

3. **Encargado de pruebas (Tester)**. Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.
4. **Encargado de seguimiento (Tracker)**. Proporciona realimentación al equipo. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar futuras estimaciones. Realiza el seguimiento del progreso de cada iteración.
5. **Entrenador (Coach)**. Es responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.
6. **Consultor**. Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas.
7. **Gestor (Big Boss)**. Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

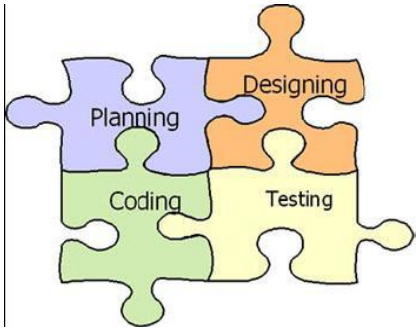

Figura: 1.3.2.1.1 Metodología XP

Además la metodología XP se basa en:

1. **Pruebas Unitarias:** pruebas realizadas a los principales procesos, de tal manera que si se quiere adelantar en algo hacia el futuro, se puedan hacer pruebas de las fallas que pudieran ocurrir. Es como adelantarse a obtener los posibles errores.
2. **Re fabricación:** se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
3. **Programación en pares:** una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento.

Esta metodología no es la apropiada para el desarrollo del proyecto Sistema de Gestión Fiscal en el cual se incluye el sistema informático integrado que se propone con este trabajo, ya que resulta demasiado práctica y rápida para un proyecto que requiere de tiempo y se caracteriza por tener procesos de negocios complejos. Los clientes no se encuentran integrados en su totalidad al proceso de desarrollo, pues la interacción directa entre clientes y desarrolladores resulta

limitada. Los requisitos de este sistema son precisos y concretos. Esto tiene contradicción con la metodología XP donde el cliente es parte importante en el desarrollo del software y se le aplica a proyectos con requisitos imprecisos y muy cambiantes.

1.3.2.2 Rational Unified Process (RUP).

El Proceso Unificado Racional (*Rational Unified Process*, RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Esta metodología es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo y trabajo de muchas metodologías utilizadas por los clientes. Fue creado por Jacobson, Rumbaugh y Booch la versión que se ha estandarizado vio la luz en 1998 y se conoció en sus inicios como Proceso Unificado de Rational 5.0; de ahí las siglas con las que se identifica a este proceso de desarrollo.

Como RUP es un proceso, en su modelación define como sus principales elementos: [6]

- **Trabajadores (“quién”):** Define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.
- **Actividades (“cómo”):** Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.
- **Artefactos (”qué”):** Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.
- **Flujo de actividades (“Cuándo”):** Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

RUP es uno de los procesos más generales de la actualidad y está tratando de adaptarse al desarrollo de cualquier proyecto aunque no sea de software. Un proyecto realizado bajo la metodología RUP se divide en cuatro fases. [6]

1. **Inicio** (determinar la visión del proyecto).
2. **Elaboración** (determinar la arquitectura óptima).
3. **Construcción** (llevar a obtener la capacidad operacional inicial).
4. **Transición** (fin del proyecto y puesta en producción).

Cada FASE necesita determinados artefactos para cumplimentar su objetivo. **HITOS**. Ver Figura1.3.2.2.1

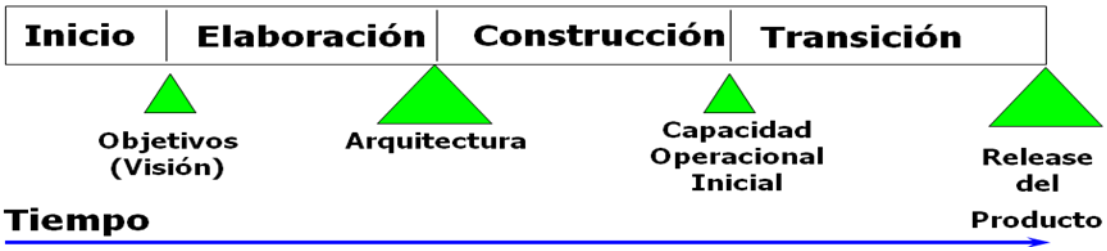

Figura: 1.3.2.2.1 Fases e Hitos
Representa un ciclo de desarrollo en la vida de un producto de software.

En RUP se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo. Ver Figura1.3.2.2.2

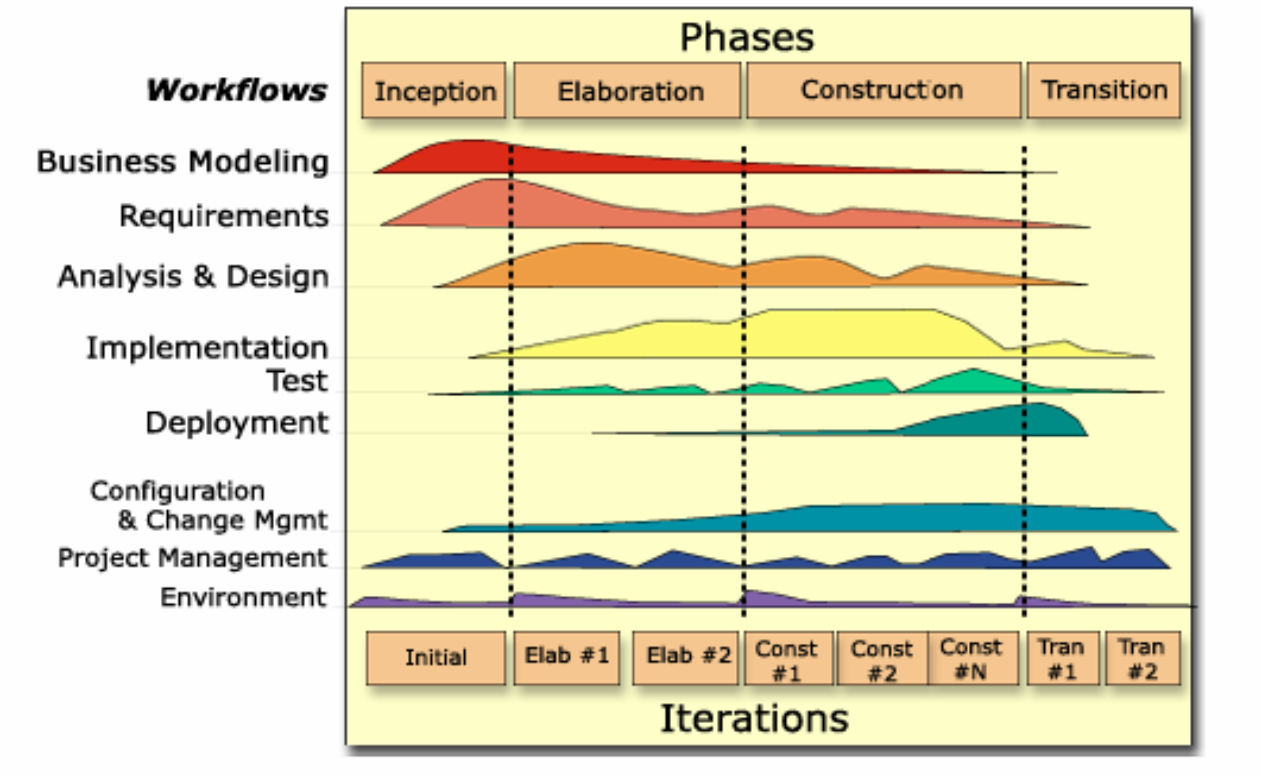

Figura: 1.3.2.2.2 Fases e iteraciones de RUP

RUP define nueve actividades a realizar en cada fase del proyecto

1. Modelado del negocio.
2. Análisis de requisitos.
3. Análisis y diseño.

4. Implementación.
5. Test.
6. Distribución.
7. Gestión de configuración y cambios.
8. Gestión del proyecto.
9. Gestión del entorno.

Y el flujo de trabajo (workflow) entre ellas, en base a los diagramas de actividad (Ver figura 1.3.2.2.3).

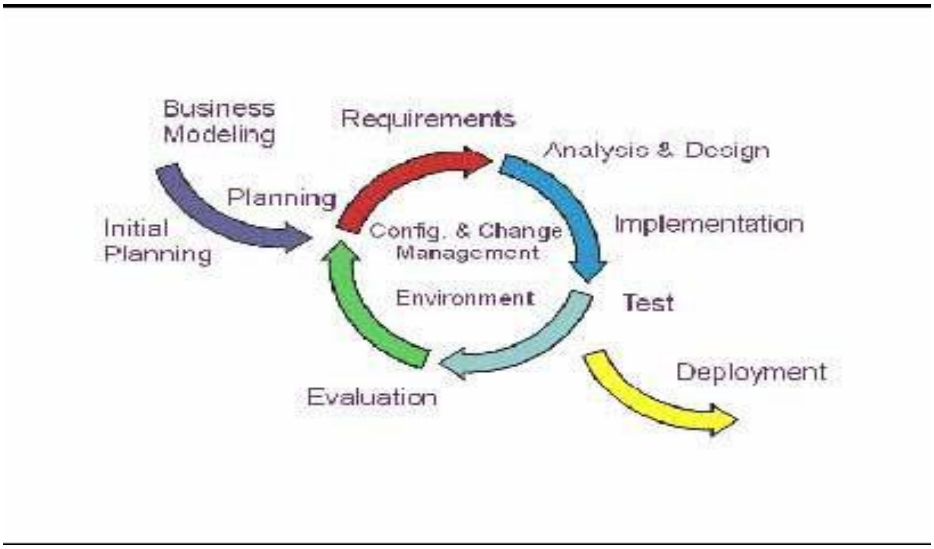

Figura 1.3.2.2.3 Flujos de trabajo de RUP

El ciclo de vida de RUP se caracteriza por: [6]

1. **Dirigido por casos de uso:** Los casos de uso reflejan lo que los usuarios finales necesitan y desean, lo cual se capturan cuando se modela el negocio, estos se representa mediante los requerimientos. En todo el proceso los casos de uso guían el desarrollo y los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso, que no es más que la descripción de los mismos.
2. **Centrado en la arquitectura:** La arquitectura muestra la visión común del sistema completo con la que el equipo del proyecto y los usuarios deben estar de acuerdo, ya que describe los elementos del modelo que son importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.
3. **Iterativo e Incremental:** RUP contiene cuatro fases con nueve flujos de trabajo para todo el desarrollo y propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto. Cada iteración se realiza de forma

planificada, por eso se dice que son mini proyectos, debido a que el trabajo se va fragmentando por pequeñas partes para facilitar su desarrollo.

RUP es una metodología usada para el desarrollar procesos pesados, ya que este tipo de proceso involucra gran cantidad de especialistas, debido al tiempo que lleva desarrollarlo. Los grandes negocios encajan perfectamente con esta metodología por el gran nivel de robustez y organización que tiene entre todas sus actividades y sus diferentes fases, por toda esta organización es que se seleccionó esta metodología ya que el producto que se pretende hacer con esta investigación es grande y tiene involucrado a una gran cantidad de especialistas lo cual se generaran varias iteraciones para lograr obtener un producto que cumpla con las exigencias del cliente y de la sociedad. Los aspectos tratados hasta el momento evidencian lo conveniente de usar esta metodología para una captura de requisitos robusta y correcta.

1.3.3 Tendencias de los Lenguajes de Programación.

Los primeros Lenguajes de Programación surgieron de la idea de Charles Babagge, a mediados del Siglo XIX desde entonces hasta la actualidad estos han evolucionado con gran rapidez al pasar de los años. La evolución de los lenguajes de programación ha estado guiada por la evolución de:

- Los ordenadores y sus sistemas operativos.
- Las aplicaciones.
- Los métodos de programación.
- Los fundamento teóricos.
- La importancia dada a la estandarización.

Ya que ellos por si solo no habrían sobre vivido al cambio sin las distintas aplicaciones que lo soportaran. A principios de la década del 50 con el fin de facilitar la labor de los programadores aparecen los lenguajes ensambladores que sustituyen los códigos de operaciones numéricos del lenguaje de máquina por símbolos alfabéticos, siguieron surgiendo nuevas necesidades y requerimientos y aparecieron los lenguajes de alto nivel, los cuales se pueden clasificar en declarativos o imperativos. Toda esta evolución les permitió a los programadores ir desarrollando lenguajes y aplicaciones cada vez mejor adaptadas a las necesidades.
Los lenguajes de programación actuales coinciden en una tendencia clara hacia las comunicaciones, ya sea de usuario a usuario ó de usuario a maquina. Internet se ha convertido en el objetivo claro de todas las empresas y usuarios particulares. Teniendo en cuenta que la programación esta orientada a dar servicio a estas dos áreas, que son las que más dinero mueven en el mundo, se ve como los lenguajes de programación e Internet están cada día más ligados y enfocados en mejorar su desarrollo para facilitar el uso de estas áreas.
Una tendencia marcada desde hace algunos años, es la dependencia creciente de componentes de software reusable por parte de programadores y equipos de desarrollo. El enfoque de la programación orientada a objetos es muestra de esto, así como su incorporación en los lenguajes

de programación. Por eso es muy probable incluso que lleguen a existir proveedores de objetos ó componentes de software, que los ofrezcan los programadores, como en la actualidad se ofrecen componentes de hardware. Entonces el enfoque del trabajo de los desarrolladores de software cambiará, tal vez, dividiéndolos en dos grupos. Todas estas tendencias están centradas en que exista una disponibilidad de lenguajes de programación con medios cómodos suficientes para que los desarrolladores sean capaces de aplicar su creatividad, ingenio y experiencia.

Algunos de los Lenguajes de Programación más populares son:

C++: Fue diseñado a principios de los años 1980, por Bjarne Stroustrup, como extensión del lenguaje de programación C, sus principales características son: [33]

- El soporte para programación orientada a objetos.
- Puede manipular directamente la memoria de la computadora.
- Tienen pocas verificaciones automáticas.
- No da soporte para interfaces de implementación.
- Tiene Entornos de desarrollos (IDE) que son comunes y le dan soporte a este lenguaje. Visual C++ .NET de Microsoft, GCC para software libre, Borland C++ Builder de Borland.

Java: Es un lenguaje orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. A diferencia de los lenguajes de programación convencionales, que generalmente están diseñados para ser compilados a código nativo, Java es compilado en un bytecode que es interpretado (usando normalmente un compilador JIT), por una máquina virtual Java. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos mucho más simple y elimina herramientas de bajo nivel como punteros, da soporte a interfaces de implementación, y posee un modelo de objetos mejor definido que C++. [32]

C#: Es un Lenguaje de Programación dentro de la plataforma Microsoft .Net es orientado a objetos, desarrollado y estandarizado por Microsoft como parte de su plataforma .NET. Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET el cual es similar al de Java aunque incluye mejoras derivadas de otros lenguajes, como Object Pascal más conocido por su IDE(Delphi). [25]

PHP5: PHP es un acrónimo recursivo que significa "**P**HP **H**ypertext **P**re-processor" (inicialmente PHP Tools, o, *P*ersonal *H*ome *P*age Tools).
Es un lenguaje de programación interpretado usado normalmente para la creación de páginas web dinámicas. En la actualidad también se puede utilizar para la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTK+.
PHP tiene algunas buenas características como son: [31]

- Es multiplataforma
- Presenta capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad.

- Permite las técnicas de Programación Orientada a Objetos.
- Permite el uso fácil y la lectura de XML de forma sencilla.
- Es libre, por lo que es una alternativa de fácil acceso para todos.

Por todas estas características fue el lenguaje elegido para desarrollar el sistema ya que también forma parte de los lenguajes más estables y usados para desarrollar aplicaciones de este tipo.

1.3.4 Herramientas Case.

Las Herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador), son aplicaciones informáticas que le facilitan el trabajo a los analistas, ingenieros de software, desarrolladores y a todos los involucrados en el proceso de desarrollo, ya que permite que se modele el sistema a realizar, ayudando de esta manera a detectar mejor lo errores y a lograr una mayor organización del proceso de desarrollo. Estas herramientas contribuyen de manera directa en todos los aspectos del ciclo de vida de desarrollo del software, en tareas como la de realizar el diseño del proyecto, los cálculo de costes y tiempo, la implementación de parte del código automáticamente generadas con un diseño dado y en la creación de toda la documentación del proyecto de una forma bien organizada.
Todos estos aspectos unidos al uso de una metodología de desarrollo son los que garantizan que las diferentes Herramientas Case persigan los siguientes objetivos:

- Mejorar la productividad en el desarrollo y mantenimiento del software.
- Aumentar la calidad del software.
- Mejorar el tiempo y coste de desarrollo y mantenimiento de los sistemas informáticos.
- Mejorar la planificación de un proyecto.
- Aumentar la biblioteca de conocimiento informático de una empresa ayudando a la búsqueda de soluciones para los requisitos.
- Automatizar varios aspectos como: desarrollo del software, documentación, generación de código, pruebas de errores y gestión del proyecto.
- Ayuda a la reutilización del software, portabilidad y estandarización de la documentación.
- Gestión global en todas las fases de desarrollo de software con una misma herramienta.

Algunas de estas Herramientas son:

1.3.4.1 Rational Rose

Rational Rose es una herramienta de desarrollo producida por IBM basada en modelos que se integra con las bases de datos y los IDE de las principales plataformas del sector. Entre sus características principales se pueden mencionar las siguientes: [11]

- Herramienta propietaria.
- Perteneciente a la familia Rational Rose.
- Madura, bien establecida en el mercado.
- Incluye una Modelación Añadida de la Web que proporciona visualización, modelación y herramientas para el desarrollo de aplicaciones Web.

- Ofrece habilidades de análisis de calidad de código, con capacidades de sincronización, así como manejo más granular y uso de modelos.
- Permite generar código a partir de modelos Ada, ANSI C++, C++, CORBA, Java/J2EE, Visual C++ y Visual Basic.
- Todos sus productos dan soporte a Unified Modeling Language (UML).
- Da soporte al Proceso Unificado de Rational (RUP).

Rational Rose presenta facilidades como:

- Diseño dirigido por modelos que ayudan a una mayor productividad de los desarrolladores, admitiendo UML, COM, OMT y Booch.
- Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Desarrollo iterativo.
- Ingeniería de ida y vuelta.

Otra de cualidades que tiene el Rational Rose es la forma que tiene de representar los sistemas desde distintos puntos de vista ya que los agrupa de la forma siguiente:

- **Casos de Uso:** Define la interacción entre actores y casos de uso. Los diagramas que se realizan son de casos de uso, de colaboración, de secuencia y de actividad.
- **Lógica:** Define fundamentalmente las clases del sistema y sus relaciones. Los diagramas principales son de clases (estático) y de estados (dinámicos).
- **Componentes:** Contiene información sobre ficheros, ejecutables y librerías del sistema. Acepta diagramas de componentes.
- **Despliegue:** Muestra la asignación de procesos al hardware. Sólo admite diagramas de despliegue.

Rational Rose es una herramienta muy potente para realizar proyectos grandes que requieran bastante tiempo y esfuerzo, está muy bien acoplado a Unified Modeling Language (UML) y con el Proceso Unificado de Rational (RUP) pero es una herramienta privativa por lo que se convierte en una limitante para su uso en esta investigación.

1.3.4.2 Visual Paradigm

Visual Paradigm es una herramienta CASE profesional que soporta la última versión de UML 2.1 así como el ciclo de vida completo del desarrollo de software, análisis y diseño orientados a objetos, construcción, pruebas y despliegue, exportación desde Rational Rose, exportación/importación XML, generación de informes y edición de figuras. Visual Paradigm tiene

disponible distintas versiones: Enterprise, Professional, Standard, Modeler, Personal y Community (que es gratuita). Algunas de sus características son: [10]

- Ofrece entorno de creación de diagramas para UML 2.1.
- Disponibilidad en múltiples plataformas.
- Disponibilidad de integrarse en los principales IDEs.
- Soporta una gama de lenguajes en la Generación de Código e Ingeniería Inversa en Java, C++, CORBA IDL, PHP, Esquema de XML, Ada y Python.
- La Generación de Código soporta C #, VB .NET, Lenguaje de Definición de Objeto (ODL), Flash Action Script, Delphi, Perl, Objetivo-C, y Ruby.
- Se integra con las herramientas Java (Eclipse/IBM WebSphere, JBuilder, NETBeans IDE, Oracle JDeveloper, BEA Weblogic).

Además de estas características ofrece aspectos importantes para el desarrollo como son:

- Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.
- El uso de un lenguaje estándar común para todo el equipo de desarrollo, lo cual facilita la comunicación entre sus miembros.
- También tiene la capacidad de ingeniería directa e inversa en la versión profesional.
- Los modelos y códigos permanecen sincronizados en todo el ciclo de desarrollo del proyecto.
- Disponibilidad de integrarse en los principales IDEs5.
- Disponibilidad en múltiples plataformas. Ingeniería Inversa Java, C++, Esquemas XML, XML6, NET exe/dll, CORBA IDL. Ingeniería de ida y vuelta.

Visual Paradigm es una herramienta fuerte para realizar trabajos con proyectos de gran tamaño al igual que el Rational, además se adapta bien al la línea de desarrollo que tiene el proyecto de Sistema de Gestión Fiscal, aunque es privativa es una herramienta multiplataforma, por lo que se escogió esta herramienta para realizar el modelado del sistema.

1.3.5 Frameworks de Desarrollo.

Los Frameworks son Arquitecturas de Software bien definidas que ayudan a organizar y a unir los diferentes componentes de un proyecto con la idea de facilitar su desarrollo.
Los frameworks en la actualidad se han convertido en piedra angular de la moderna ingeniería del software. El desarrollo del framework está ganando rápidamente la aceptación entre los diferentes desarrolladores, debido a su capacidad para promover la reutilización del código del diseño y el código fuente. A continuación se señalan algunos de los frameworks que se utilizan en la actualidad para el desarrollo de sistemas.

1.3.5.1 Zend Framework

Es un framework utilizado para desarrollar aplicaciones y servicios Web con PHP, que brinda soluciones para construir sitios web con PHP5. Es orientado a objeto, Tiene una estructura de componentes bien diseñada con pocas dependencias entre ellos. Presenta las siguientes características: [14]

- Trabaja con MVC (Model View Controller) y PHP5.
- Es Open Source.
- Cuenta con módulos para manejar archivos PDF, canales RSS, Web Services (Amazon, Flickr, Yahoo).
- Incluye objetos de las diferentes bases de datos.
- Completa documentación y tests de alta calidad.
- Robustas clases para autenticación y filtrado de entrada.
- Clientes para servicios web, incluidos Google Data APIs y StrikeIron.
- Muchas otras clases útiles para hacerlo tan productivo como sea posible.

Zend Framework es nuevo en su desarrollo la ultima versión hasta el momento es Zend Framework 1.7 que ya se encuentra liberado y trae muchas novedades como son:[14]

- Zend_Amf soportando los protocolos AMF0 y AMF3. Sirve para interactuar con Adobe Flex-Air.
- ZendX_JQuery. Soporte jQuery a través de la librearía de extras de Zend (ZendX).
- Soporte para documentos Open Office XML en el indexador de Zend_Search_Lucen.
- Mejoras en el rendimiento de Zend_Loader, Zend_Controller, y los componentes de servidor.
- Mejoras en Zend_Mail_Storage_Writable_Maildir para el envío de mails.
- Zend_Text_Table que crea tablas sin <table>, a base de caracteres de texto.
- Soporte para Zend_Db_Table_Select en Zend_Páginator.
- Posibilidad de usar rutas encadenadas para Hostname-Routes vía Zend_Config.
- Mejoras en la internacionalización.
- Mejoras en el envío de ficheros.
- Soporte para adapters personalizados en Zend_Páginator.

Aunque presenta muchas mejoras en su última versión al Zend Framework le falta madurar mas para convertirse en unos de los framework más usados en el mundo, su curva de aprendizaje es elevada por el uso del lenguaje YAML, pero se esta constantemente trabajando por parte de los desarrolladores de Zend, para que ocupe una posición importante en el desarrollo de las diferentes aplicaciones.

1.3.5.2 Spring .NET

Spring .NET es un framework para realizar aplicaciones web para .NET basado en el framework Spring de Java. Es de código abierto, Fue liberado bajo la Apache License 2.0 para el desarrollo de aplicaciones que facilita la construcción de aplicaciones empresariales en .NET. Tiene componentes basados en patrones de diseño ya probados y que pueden ser integrados en todas las capas de la arquitectura de una aplicación. [15]
Está compuesto de los siguientes módulos:

- **Spring.Core:** Es el módulo fundamental y proporciona los servicios básicos en los que se asienta el resto, así como los *patrones* básicos en los que se basa.
- **Spring.Aop:** Módulo para la programación orientada a aspectos.
- **Spring.Web:** Extiende ASP.NET con distintas funcionalidades, como soporte extendido de localización.
- **Spring.Services:** Permite acceder a cualquier objeto "normal" como un servicio COM+ u objeto remoto.
- **Spring.Data y Spring.ORM**: Abstracción de datos que se puede usar con varios proveedores de datos, desde ADO.NET a varios ORM.

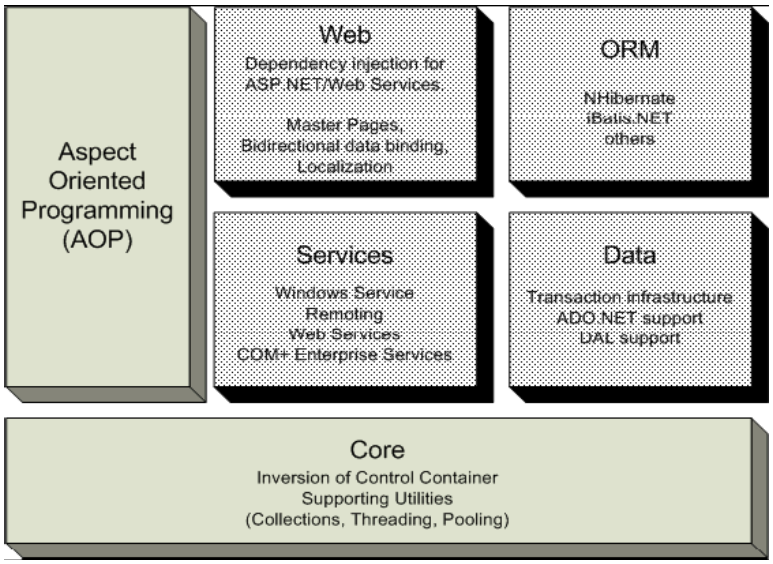

Figura: 1.3.5.2.1 Spring.NET: Componentes Generales.

En esta versión para la plataforma .NET, añade funcionalidades que reducen drásticamente la cantidad de código escrito por parte de los desarrolladores. Incorpora técnicas ya utilizadas en otras plataformas de desarrollo de aplicaciones empresariales y también permite la integración con otros frameworks open source, como NHibernate, IBatis.NET. Spring.net no se orienta únicamente a una parte de una aplicación ya que influye sobre todas las capas de la misma, no

se puede considerar un framework invasivo ya que puede ser fácilmente reemplazado. Aunque Spring brinda muchas facilidades también tiene deficiencias como:

- Su curva de aprendizaje es muy escarpada, debido a los nuevos conceptos de programación que implementa, ya que contiene una abrumadora cantidad de opciones a configurar en ficheros XML planos, que pueden convertirse en un problema para quien se acerque por primera vez a las herramientas.
- La principal desventaja es que al tener las dependencias en un fichero XML, aumenta el tiempo de respuesta de la aplicación, puesto que en este momento el framework interpreta todas las instancias que tiene configuradas en el fichero.

1.3.5.3 Symfony

Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web.
Algunas de sus características son las siguientes: [16]

- Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web.
- Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja.
- Automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.
- Es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft.
- Es multiplataforma, se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows.

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony automatiza la mayoría de elementos comunes de las aplicaciones web, como por ejemplo: [15]

- La capa de internacionalización, que permite la traducción de los datos y de la interfaz, así como la adaptación local de los contenidos.
- La capa de presentación utiliza plantillas y layouts que pueden ser creados por diseñadores HTML sin ningún tipo de conocimiento del framework.
- Los helpers incluidos permiten minimizar el código utilizado en la presentación, ya que encapsulan grandes bloques de código.
- Los formularios incluyen validación automatizada y relleno automático de datos, lo que asegura la obtención de datos correctos y mejora la experiencia de usuario.
- Mecanismos de escape de datos, el cual permiten una mejor protección contra los ataques producidos por datos corruptos.
- La gestión de la caché es formidable ya que reduce el ancho de banda utilizado y la carga del servidor.

- La autenticación y la gestión de credenciales simplifican la creación de secciones restringidas y la gestión de la seguridad de usuario.
- El sistema de enrutamiento y las URL limpias permiten considerar a las direcciones de las páginas como parte de la interfaz, además de estar optimizadas para los buscadores.
- El soporte de e-mail incluido y la gestión de APIs permiten a las aplicaciones web interactuar más allá de los navegadores.
- Los listados son más fáciles de utilizar debido a la paginación automatizada, el filtrado y la ordenación de datos.
- Las interacciones con Ajax son muy fáciles de implementar mediante los *helpers* que permiten encapsular los efectos JavaScript compatibles con todos los navegadores en una única línea de código.

Aunque Symfony también posee una curva de aprendizaje elevada fue el framework escogido para desarrollar la aplicación por la gran variedad de características favorables que presenta para realizar el desarrollo de aplicaciones de este tipo y por el nivel de seguridad que este presenta. Symfony esta basado en el estilo arquitectónico Modelo-Vista-Controlador, el cual está formado por 3 niveles. Symfony implementa estos niveles de una forma muy particular: [16]

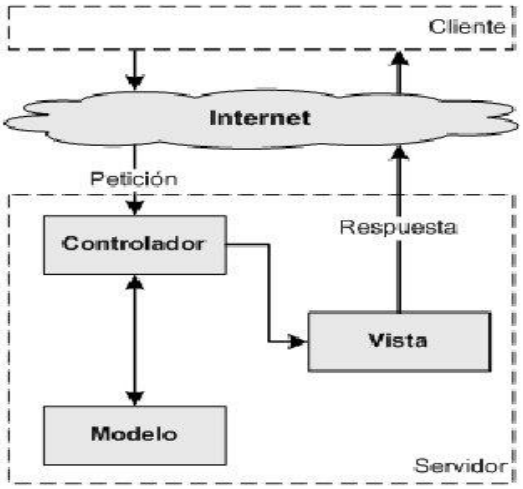

Figura: 1.3.5.3.1 Estilo Modelo-Vista-Controlador.

Modelo

El componente que se encarga por defecto de gestionar el modelo en Symfony, es una capa de tipo ORM (object relational mapping) o “mapeo de objetos a bases de datos” realizada mediante el proyecto Propel. En las aplicaciones Symfony, el acceso y la modificación de los datos almacenados en la base de datos se realiza mediante objetos; asegurando que nunca se acceda de forma explícita a la base de datos. Este comportamiento permite un alto nivel de abstracción y permite una fácil portabilidad.

La principal ventaja que aporta el ORM es la reutilización, permitiendo llamar a los métodos de un objeto de datos desde varias partes de la aplicación e incluso desde diferentes aplicaciones.

Vista

La vista se encarga de producir las páginas que se muestran como resultado de las acciones. La vista en Symfony está compuesta por diversas partes, estando cada una de ellas especialmente preparada para que pueda ser fácilmente modificable por la persona que normalmente trabaja con cada aspecto del diseño de las aplicaciones.

Los diseñadores Web normalmente trabajan con las plantillas (que son la presentación de los datos de la acción que se está ejecutando) y con el layout (que contiene el código HTML común a todas las páginas). Estas partes están formadas por código HTML que contiene pequeños trozos de código PHP, que normalmente son llamadas a los diversos helpers (funciones de PHP que devuelven código HTML) disponibles.

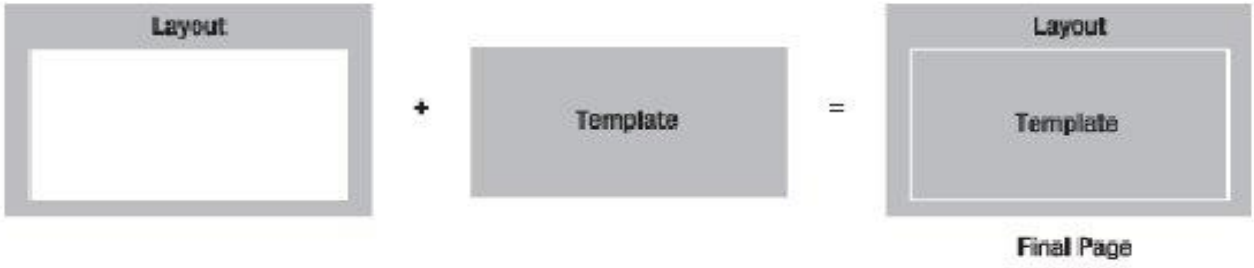

Figura 1.3.5.3.2 Vista

Controlador

En Symfony, la capa del controlador, que contiene el código que liga la lógica de negocio con la presentación, está dividida en varios componentes que se utilizan para diversos propósitos:

- El controlador frontal es el único punto de entrada a la aplicación. Carga la configuración y determina la acción a ejecutarse.
- Las acciones contienen la lógica de la aplicación. Verifican la integridad de las peticiones y preparan los datos requeridos por la capa de presentación.
- Los objetos request, response y session dan acceso a los parámetros de la petición, las cabeceras de las respuestas y a los datos persistentes del usuario. Se utilizan muy a menudo en la capa del controlador.

Los filtros son trozos de código ejecutados para cada petición, antes o después de una acción. Por ejemplo, los filtros de seguridad y validación son comúnmente utilizados en aplicaciones Web dándole mayor seguridad a la información a la hora de ser accedida por algún usuario.

1.3.6 Entorno de Desarrollo Integrado (Eclipse, Visual Studio .NET).

Un Entorno de Desarrollo Integrado o en inglés Integrated Development Environment ('IDE') es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, puede utilizarse para varios.
Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI. Los IDEs pueden ser aplicaciones por si solas o pueden ser parte de aplicaciones existentes. El lenguaje Visual Basic por ejemplo puede ser usado dentro de las aplicaciones de Microsoft Office, lo que hace posible escribir sentencias Visual Basic en forma de macros para Microsoft Word.
Los IDEs proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Java, C#, Delphi, Visual Basic, Object Pascal, Velneo.
Algunos de los IDE mas usados son Eclipse y Visual Studio .NET, a continuación se describen algunas de sus características.

1.3.6.1 Eclipse

Eclipse es una plataforma de software de Código abierto, independientemente de ser una plataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores.
Esta plataforma, ha sido usada para desarrollar un IDE, como el llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se embarca como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). Sin embargo, también se puede usar para otros tipos de aplicaciones, como es el cliente BitTorrent Azureus.
El entorno integrado de desarrollo (IDE) de Eclipse emplea módulos (plug-in) para proporcionar toda su funcionalidad al frente de la plataforma de cliente rico. Este mecanismo de módulos es una plataforma ligera para componentes de software. Adicionalmente le permite a Eclipse extenderse usando otros lenguajes de programación como son PHP, C/C++ y Phyton, le permite además trabajar con lenguajes para procesado de texto como LaTeX, aplicaciones en red como Telnet y Sistema de gestión de base de datos.
La arquitectura plugin permite escribir cualquier extensión deseada en el ambiente, como Gestión de la configuración. Se provee soporte para Java y CVS en el SDK de Eclipse. Este no debe ser usado solamente para soportar otros lenguajes de programación ya que tiene también otras utilidades como por ejemplo a través de "plugins" libremente disponibles es posible añadir:

- Control de versiones con Subversión, vía Subclipse.
- Herramienta Gestor de Base de Datos, PostgreSQL 8.1, PgAdmin III.
- Herramienta de Gestión de Proyecto, Dotproject.
- Herramienta de control de versiones, Subversión, RapidSvn.
- Herramienta para el trabajo con imágenes, Gimp.
- Herramienta para el trabajo de Réplica con PostgreSql, PgCluster (Replicación

Multimaestro).

Por estas facilidades que brinda Eclipse fue el IDE escogido para realizar el proyecto, la versión que fue escogida es la Eclipse 3.2.2 que además tiene las características siguientes: [24]

- Editor de Texto.
- Resaltado de sintaxis.
- Compilación en tiempo real.
- Pruebas unitarias con JUnit.
- Control de versiones con CVS.
- Asistentes (wizards): para creación de proyectos, clases, tests, etc.
- Refactorización

1.3.6.2 Visual Studio .NET

Visual Studio .NET no es una plataforma de software de Código abierto, es un Entorno de Desarrollo Integrado (IDE) diseñado para facilitar la construcción y el desarrollo de servicios Web XML, aplicaciones para Windows y soluciones Web. Entre sus múltiples características cabe destacar su soporte multi-lenguaje, que permite integrar en una misma aplicación código escrito en diferentes lenguajes de programación. Es el único entorno de programación creado exclusivamente para servicios Web XML. Al permitir que las aplicaciones se comuniquen y compartan datos a través de Internet. Visual Studio .NET y los servicios Web XML proporcionan un modelo sencillo y flexible basado en estándares que permite a los programadores ensamblar aplicaciones a partir de código nuevo o existente, independientemente de la plataforma, del lenguaje de programación o del modelo de objetos. Visual Studio se caracteriza por: [27]

- Soporte para XML Web Services.
- Diseño de interfaces mediante Windows Forms.
- Diseño web mediante Web Forms o Mobile Web Forms.
- Creación de aplicaciones para Pocket PC-Windows CE .NET.
- Integración con .NET Framework y el Common Language Runtime (CLR)
- Asistente para actualización de código de VB 6.0 a VB.NET.
- Creación de aplicaciones para consola.
- Plantillas de Bibliotecas de Clases.
- Librerías para controles Windows y controles Web.
- Creación de Servicios Windows ejecutables al inicio del sistema.
- Otras características a destacar son: un entorno integrado de desarrollo, un completo depurador, Crystal Reports o un asistente para el diseño HTML.
- Soporte Multilenguaje (Visual Basic .NET, Visual C# .NET, Visual C++ .NET, Visual J#. Net).

Visual Studio .NET es muy usado en el mundo para realizar aplicaciones robustas pero no cumple con la característica principal que exige el Proyecto de Gestión Fiscal ya que el principal objetivo es desarrollar en herramientas de código abierto y además al discutirse con los clientes también coincidieron en el uso de tecnología libre.

1.3.7 Sistemas de Gestión de Base de Datos.

Hoy en día los SGBD (Sistema de Gestión de Base de Datos) facilitan el uso de técnicas para gestionar convenientemente la información almacenada, con una forma fácil de interpretar y útil para el usuario, con facilidad y fiabilidad. Es por ello que se han convertido en el instrumento o soporte básico más ampliamente usado en la gestión de los sistemas informáticos.

1.3.7.1 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, basado en el proyecto POSTGRES, de la universidad de Berkeley. Es un servidor de base de datos objeto relacional libre, liberado bajo la licencia BSD. Como muchos otros proyectos open source, el desarrollo de PostgreSQL no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo, dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group).
Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido, más tarde en otros sistemas de gestiones comerciales. PostgreSQL es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. A pesar de esto, PostgreSQL no es un sistema de gestión de bases de datos puramente orientado a objetos.
Algunas características de PostgreSQL son: [20]

- Corre en casi todos los principales sistemas operativos: Linux, Unix, BSDs, Mac OS, Beos, Windows, etc.
- Soporte nativo para los lenguajes más populares: PHP, C, C++, Perl, Python, etc.
- Soporte de protocolo de comunicación encriptado por SSL.
- Extensiones para alta disponibilidad, nuevos tipos de índices, datos espaciales, minería de datos, etc.

PostgreSQL también otras características como son:

- Implementación del estándar SQL92/SQL99.
- Soporta distintos tipos de datos, además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP…), cadenas de bits, etc.
- Incorpora una estructura de datos array.
- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, etc.

- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.
- Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.

Por todas estas características y su compatibilidad con los distintos lenguajes de programación y sistemas operativos, por ser libre y tener una amplia comunidad de desarrollo. PostgreSQL fue escogido para realizar la base de datos del proyecto Sistema de Gestión Fiscal, pues cumple con los requisitos que pide el proyecto para realizar el desarrollo del software.

1.3.7.2 Microsoft SQL Server

Es un sistema de gestión de bases de datos relacionales desarrollado por Microsoft. Para el desarrollo de aplicaciones más complejas (tres o más capas), Microsoft SQL Server incluye interfaces de acceso para la mayoría de las plataformas de desarrollo, incluyendo .NET, además de ser uno de los mejores sistemas de gestión de bases de datos relacionales (SGBD) basada en el lenguaje SQL, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea. Las necesidades y requerimientos de los clientes han llevado a la creación de innovaciones de producto significativas para facilitar la utilización, escalabilidad, confiabilidad y almacenamiento de datos. [34]

Ventajas:

- Soporta la configuración automática y la auto-optimización.
- Administración multiservidor para un gran número de servidores.
- Gran variedad de opciones de duplicación de cualquier base de datos.
- Acceso universal a los datos (Universal Data Access).
- Fácil de usar.
- Escalabilidad
- Múltiples instancias y Failover.
- Integración con la Web.
- Acceso Web a datos.
- Soporte para XML (cláusula FOR XML).

Desventajas:

Licencias con costos altos.
Plataformas Windows.

1.3.8 Patrones de Diseño.

Los patrones de diseño (design patterns del inglés), son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son propuestas basadas en la experiencia en las cuales se ha demostrado que funcionan. Los patrones de diseño identifican: Clases, Instancias, Roles, Colaboraciones y la distribución de responsabilidades.
Los patrones de diseño no son fáciles de entender, pero una vez que se logra conocer su funcionamiento el diseño será mucho más flexible, modular y reutilizable. Han revolucionado el diseño orientado a objetos y todo buen arquitecto de software debería conocerlos.
Los patrones de diseño pretenden:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

Los patrones según su propósito se pueden clasificar en:

- **Patrones de Creación:** Creación de instancias de objetos.
- **Patrones Estructurales:** Relaciones entre clases, combinación y formación de estructuras mayores.
- **Patrones de Comportamiento:** Interacción y cooperación entre clases.

Los patrones de diseño más habituales según el libro "Design Patterns" son: [9]
Patrones de creación

- **Abstract Factory**. Proporciona una interfaz para crear familias de objetos o que dependen entre sí, sin especificar sus clases concretas.
- **Builder**. Separa la construcción de un objeto complejo de su representación, de forma que el mismo proceso de construcción pueda crear diferentes representaciones.
- **Factory Method**. Define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar. Permite que una clase delegue en sus subclases la creación de objetos.
- **Prototype**. Especifica los tipos de objetos a crear por medio de una instancia prototípica, y crear nuevos objetos copiando este prototipo.
- **Singleton**. Garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella.

Patrones estructurales

- **Adapter**. Convierte la interfaz de una clase en otra distinta que es la que esperan los clientes. Permiten que cooperen clases que de otra manera no podrían, por tener interfaces incompatibles.
- **Bridge**. Desvincula una abstracción de su implementación, de manera que ambas puedan variar de forma independiente.
- **Composite**. Combina objetos en estructuras de árbol para representar jerarquías de parte-todo. Permite que los clientes traten de manera uniforme a los objetos individuales y a los compuestos.
- **Decorator**. Añade dinámicamente nuevas responsabilidades a un objeto, proporcionando una alternativa flexible a la herencia para extender la funcionalidad.
- **Facade**. Proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar.
- **Flyweight**. Usa el compartimiento para permitir un gran número de objetos de grano fino de forma eficiente.
- **Proxy**. Proporciona un sustituto o representante de otro objeto para controlar el acceso a éste.

Patrones de comportamiento

- **Chain of Responsibility**. Evita acoplar el emisor de una petición a su receptor, al dar a más de un objeto la posibilidad de responder a la petición. Crea una cadena con los objetos receptores y pasa la petición a través de la cadena hasta que esta sea tratada por algún objeto.
- **Command**. Encapsula una petición en un objeto, permitiendo así parametrizar a los clientes con distintas peticiones, en colar o llevar un registro de las peticiones y poder deshacer las operaciones.
- **Interpreter**. Dado un lenguaje, define una representación de su gramática junto con un intérprete que usa dicha representación para interpretar las sentencias del lenguaje.
- **Iterator**. Proporciona un modo de acceder secuencialmente a los elementos de un objeto agregado sin exponer su representación interna.
- **Mediator**. Define un objeto que encapsula cómo interactúan un conjunto de objetos. Promueve un bajo acoplamiento al evitar que los objetos se refieran unos a otros explícitamente, y permite variar la interacción entre ellos de forma independiente.
- **Memento**. Representa y externaliza el estado interno de un objeto sin violar la encapsulación, de forma que éste puede volver a dicho estado más tarde.
- **Observer**. Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos.
- **State**. Permite que un objeto modifique su comportamiento cada vez que cambia su estado interno. Parecerá que cambia la clase del objeto.

- **Strategy**. Define una familia de algoritmos, encapsula uno de ellos y los hace intercambiables. Permite que un algoritmo varíe independientemente de los clientes que lo usan.
- **Template Method**. Define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos. Permite que las subclases redefinan ciertos pasos del algoritmo sin cambiar su estructura.
- **Visitor**. Representa una operación sobre los elementos de una estructura de objetos. Permite definir una nueva operación sin cambiar las clases de los elementos sobre los que opera.

1.3.9 Patrones GRASP

GRASP es un acrónimo que significa “patrones generales de software para asignar responsabilidades” (General Responsibility Asignment Software Patterns). El nombre se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos. Estos describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.
Ejemplo de patrones GRASP son:

- Experto.
- Creador.
- Alta Cohesión.
- Bajo Acoplamiento.
- Controlador.
- Polimorfismo.
- Fabricación Pura.
- Indirección.
- No Hables con Extraños (Ley de Demeter).

A continuación se especifican de los patrones GRASP básicos, la solución, el problema y los beneficios.

Experto: [7]
Solución: Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.
Problema: ¿Quién asumirá la responsabilidad en el caso general?
Beneficios:

- Se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento.
- El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clases "sencillas” y más cohesivas que son más fáciles de comprender y de mantener. Así se brinda soporte a una alta cohesión.

Creador [7]
Solución: Asignar a la clase B la responsabilidad de crear una instancia de clase A, si se cumple una de las siguientes condiciones:
1. B contiene A.
2. B agrega A.
3. B tiene los datos de inicialización de A.
4. B registra A.
5. B utiliza A muy de cerca.

Problema: ¿Quién crea?
Beneficios:

- Brinda soporte a un bajo acoplamiento.
- Supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización.
- La clase *creada* tiende a ser visible a la clase *creador.*

Alta Cohesión [7]
Solución: Asignar las responsabilidades de modo que se mantenga una alta cohesión.
Problema: ¿Como mantener controlable la complejidad?
Beneficios:

- Mejoran la claridad y la facilidad con que se entiende el diseño.
- Se simplifican el mantenimiento y las mejoras en funcionalidad.
- A menudo se genera un bajo acoplamiento.
- La ventaja de una gran funcionalidad soporta una mayor capacidad de reutilización, porque una clase muy cohesiva puede destinarse a un propósito muy específico.

Bajo acoplamiento [7]
Solución: Asignar las responsabilidades de modo que se mantenga bajo acoplamiento.
Problema: ¿Como dar soporte a poca dependencia y a una mayor reutilización?
Beneficios:

- No se afectan por cambios de otros componentes.
- Fáciles de entender por separado.
- Fáciles de reutilizar.

Controlador [7]
Solución: Asignar la responsabilidad de administrar un mensaje de eventos del sistema a una clase que represente una de las siguientes opciones:

- El negocio o la organización global (un controlador de fachada).
- El "sistema" global (un controlador de fachada).
- Un ser animado del dominio que realice el trabajo (un controlador de papeles).

- Una clase artificial (Fabricación Pura) que represente el caso de uso (un controlador de casos de uso).

Problema: ¿Quién administra un evento del sistema?
Beneficios:

- Mayor potencial de los componentes reutilizables.
- Reflexionar sobre el estado del caso de uso

1.3.10 Estilos Arquitectónicos.

Un patrón de arquitectura de software describe un problema particular y recurrente del diseño, que surge en un contexto específico, y presenta un esquema genérico y probado de su solución.
Estos estilos arquitectónicos son una generalización y abstracción de los patrones de diseño, la diferencia distintiva con respecto al nivel del diseño radica en que la descripción del nivel arquitectónico implica el uso de elementos de software que no tienen una representación directa en la mayoría de los lenguajes de programación; es decir, elementos abstractos con los cuales trabaja el arquitecto y que los programadores deberán refinar y proyectar sobre la tecnología de implementación disponible. Resulta difícil identificar claramente entre el nivel de diseño y el nivel arquitectónico ya que no existe una notación que permita expresar siempre el nivel arquitectónico sin adentrarse en el detalle del diseño, el nivel arquitectónico es usado para describir sistemas completos que luego son detallados por partes en el diseño.
Los estilos arquitectónicos son una buena aproximación para describir la arquitectura sin describir el diseño, conforman un conjunto de reglas de diseño que identifican las clases de componentes y conectores que se pueden utilizar para componer un sistema o subsistema. Estos estilos facilitan la comunicación entre los diferentes participantes en el desarrollo de software, resaltan decisiones de diseño que pueden tener gran impacto en el proceso de desarrollo posterior. Dan una visión de como se estructura el sistema y de como trabajan juntos los diferentes componentes.
A continuación se describen algunos de los estilos arquitectónicos que existen y que deben considerarse su estudio para la construcción del sistema que se quiere implementar, con vista a realizar una buena Arquitectura de Software y facilitar así todo el proceso de desarrollo.

1.3.10.1 Tubería y filtros

Se basa en el patrón (pipe and filter) tubería y filtro, que consta de un conjunto de componentes denominados “filtros” conectados entre si por “tuberías” que permiten la trasmisión de datos de un componente al siguiente. Cada filtro trabaja independiente a los componentes que están situados antes o después de este. Estos se diseñan de tal modo que esperan un conjunto de datos de entrada en un formato determinado y obtienen un conjunto de datos de salida en un formato específico. Si el flujo degenera en una única línea de transformación se denomina secuencial batch. El estilo tubería-filtros propiamente dicho enfatiza la transformación incremental de los datos por sucesivos componentes (Ver Figura 1.3.10.1.2). [19]
Por ejemplo:

Los primeros compiladores operaban conforme a un estilo de tubería y filtro bastante puro, en ocasiones en variantes de proceso por lotes. A medida que los compiladores se tornaron más sofisticados, se fueron añadiendo elementos tales como tablas de símbolos, generalmente compartidas por varios filtros, el ejemplo que se muestra en la figura (Ver Figura 1.3.10.1.2) es un ejemplo de secuencial batch.

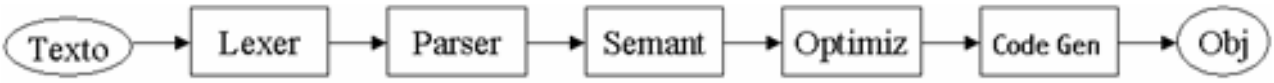

Figura 1.3.10.1.1 Compilador

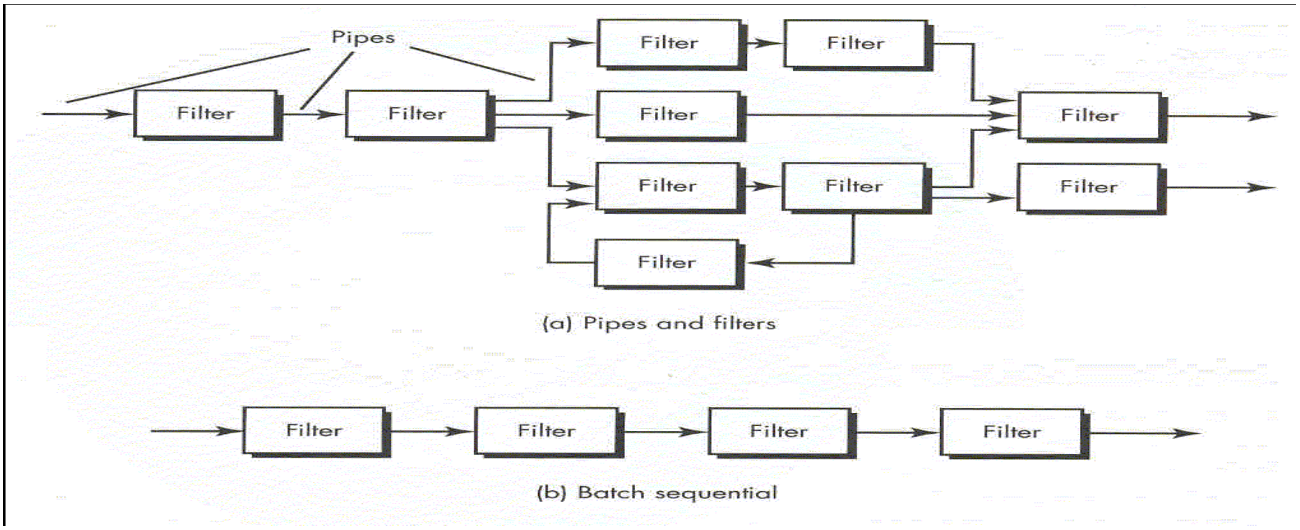

Figura 1.3.10.1.2 Tubería y Filtro

El estilo tubería y filtro a pesar de tener las siguientes ventajas:

- Permite entender el sistema global en términos de la combinación de componentes.
- Soporta de buena manera la reutilización.
- Los filtros son independientes de sus vecinos.
- Facilidad de mantenimiento y mejora.
- Soportan la ejecución concurrente.

No es aconsejable para cuando se necesita interactividad en un sistema pues presenta problemas de rendimiento ya que los datos se transmiten en forma completa entre filtros, por lo que no es el estilo apropiado para realizar el sistema propuesto.

1.3.10.2 Estilo de Llamada y Retorno.

Los estilos de Llamada y Retorno enfatizan la modificabilidad y la escalabilidad del software, son los estilos más utilizados, generalizados en los sistemas a gran escala. Dentro de esta familia se encuentran el Modelo-Vista-Controlador (MVC), la Arquitectura Orientada a Objetos y la Arquitectura en Capas.

Arquitecturas Orientadas a Objetos

En este estilo las representaciones de los datos y las operaciones están encapsuladas en un tipo abstracto de datos u objeto, los componentes son objetos, las invocaciones de métodos son los conectores. [19]
Los objetos son responsables de la integridad de sus representaciones y dicha representación es ocultada al resto de los objetos.
Entre las principales características de esta arquitectura se destacan:

- Los componentes del estilo se basan en principios Orientados a Objetos: encapsulamiento, herencia y polimorfismo. Así mismo las unidades de modelado, diseño e implementación, y los objetos y sus interacciones son el centro de las incumbencias en el diseño de la arquitectura y en la estructura de la aplicación.
- Las interfaces están separadas de las implementaciones. En general la distribución de objetos es transparente.
- Se puede modificar la implementación de un objeto sin afectar a sus clientes. Así mismo es posible descomponer problemas en colecciones de agentes en interacción. Además un objeto es ante todo una entidad reutilizable en el entorno de desarrollo.

Las limitaciones de este estilo arquitectónico radican en que para poder invocar métodos de un objeto se debe conocer su identidad, y si se modifica el objeto se deben modificar todos los objetos que lo invocan.

Arquitecturas en Capas

Este estilo esta organizado jerárquicamente en capas, donde cada capa provee servicios a la capa superior y es servido por la capa inferior, los componentes son cada una de las capas y los conectores son los protocolos de interacción entre las capas, esta iteración está limitada a la capa adyacente. [19]

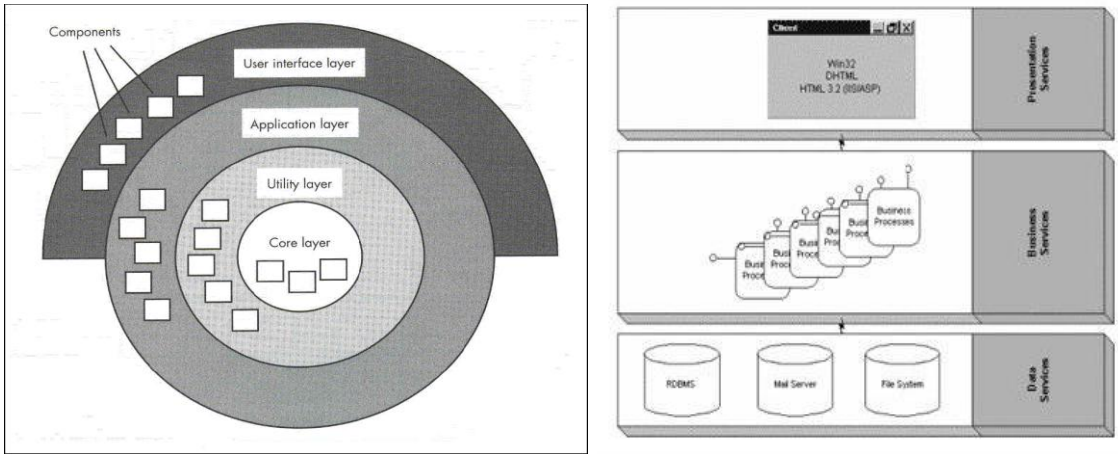

Figura 1.3.10.2.1 Arquitectura en Capas

Este estilo facilita la descomposición del problema en varios niveles de abstracción, permite la reutilización de código, además soporta fácilmente la evolución segura del sistema ya que los cambios solo afectan a las capas vecinas, esto permite que se puedan cambiar las implementaciones siempre y cuando se respeten las interfaces con las capas adyacentes, el estilo también admite muy naturalmente optimizaciones y refinamientos del proceso.
Una desventaja de este estilo está dada porque muchos problemas no admiten un buen mapeo en una estructura jerárquica, es decir no todos los sistemas pueden estructurarse en capas.

Model-View-Controller (MVC)

En ocasiones se le define más bien como un patrón de diseño o como práctica recurrente. [19]
Este patrón separa la organización del sistema en tres clases diferentes:

- **Modelo**. El modelo administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).
- **Vista**. Maneja la visualización de la información enviada por el controlador.
- **Controlador**. Interpreta las acciones del ratón y el teclado, informando al modelo y/o a la vista para que cambien según resulte apropiado.

Tanto lo vista como el controlador dependen del modelo, el cual no depende de ninguna de las otras clases, esta estructura permite que se construya y se pruebe el modelo sin importar la forma de la representación visualizar.

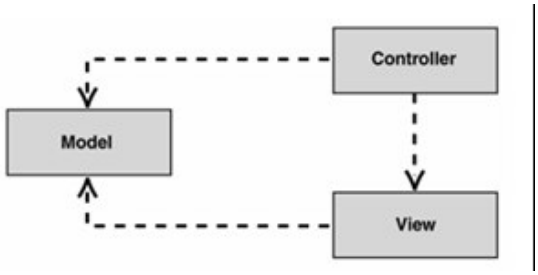

Figura 1.3.10.2.2 Modelo-Vista-Controlador

Como la vista se encuentra separada del modelo, porque no hay una dependencia directa del modelo con respecto a la vista, esto facilita que tenga soporte para múltiples vistas (es decir la misma información se puede mostrar de diferentes formas). Esta ventaja es muy importante ya que se pueden agregar nuevas opciones a la vista sin afectar el negocio, debido a que los requerimientos de la interfaz de usuario cambian con mayor frecuencia que los del negocio.

1.3.10.3 Arquitecturas de Pizarra o Repositorio

En esta arquitectura hay dos componentes principales:

- Una estructura central de datos (representa el estado actual del proceso).
- Componentes independientes (operan en función del depósito de datos).

Los subestilos característicos de la familia serían los repositorios, las bases de datos, las arquitecturas basadas en hipertextos y las arquitecturas de pizarra.
Un sistema de pizarra se implementa para resolver problemas en los cuales las entidades individuales se manifiestan incapaces de acercarse a una solución, cuando no existe una solución analítica, o cuando sí existen pero es inviable por la dimensión del espacio de búsqueda. Es utilizada en la inteligencia artificial distribuida o cooperativa, en robótica, en modelos multi-agentes, en programación evolutiva, en gramáticas complejas, etc. [19]

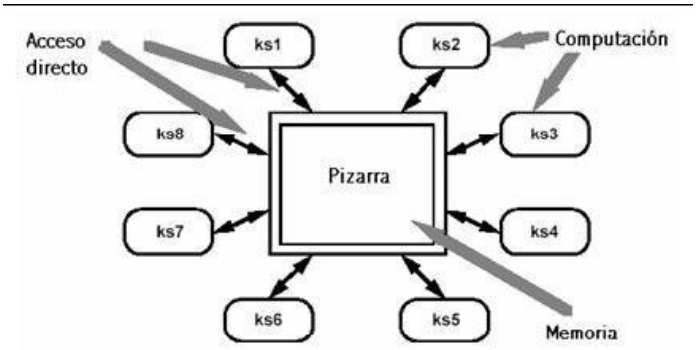

Figura 1.3.10.5.1 Pizarra

Ventajas

- Posibilita la integración de agentes.

- Adecuado para la resolución de problemas no deterministas.
- Se puede resumir el estado de conocimiento en cada momento del proceso.

Desventajas

- Estructura de datos común a todos los agentes.
- Problemas de carga a la hora de chequear y vigilar el estado de la pizarra.

1.3.11 Métricas

Las Métricas son un conjunto de medidas que existen a nivel internacional que ayudan a comprender el proceso técnico que se utiliza en el desarrollo de un producto, permiten evaluar el análisis y el diseño lo cual le da al producto un nivel más alto de calidad. A continuación algunas de las métricas asociadas al diseño.

Métricas del Modelo de Diseño

Las métricas del Modelo de Diseño le permiten al diseñador obtener una mejor visión interna del sistema, lo que ayuda a que el diseño evolucione a niveles superiores de calidad. Estas métricas están compuestas por:

- Métricas de diseño arquitectónico.
- Métricas de Diseño a nivel de componente**.**

Métricas de Diseño Arquitectónico

Las métricas de diseño arquitectónico se concentran en las características de la arquitectura del programa, con especial énfasis en la estructura arquitectónica y en la eficiencia de los módulos o clases. Están compuestas por:

- Métricas de complejidad propuestas por Card y Glass en 1990.
- **Complejidad estructural:** $S(i) = f^2_{out}(i)$, donde f_{out} (i) = expansión del módulo i (número de módulos inmediatamente subordinados al módulo i).
- **Complejidad de Datos**. Sobre la interfaz interna del módulo: $D(i) = v(i) / [f_{out}(i) + 1]$, donde v(i) es el número de variables que entran o salen del módulo.
- **Complejidad del Sistema:** $C(i) = S(i) + D(i)$.

A medida que crecen los valores de complejidad, crece la complejidad arquitectónica del sistema.

En 1981 Henry y Kafura proponen la métrica de complejidad de expansión – concentración, que considera las estructuras de datos que recoge (concentran) o actualizan (expansión):
$MHK = \text{longitud}(i) \times [f_{in}(i) + f_{out}(i)]^2$ donde longitud (i) = número de sentencias en lenguaje de programación.
En 1991, Fenton propone medidas de morfología simples basadas en la estructura de módulos jerárquicos del sistema:

- Tamaño.
- Profundidad.
- Anchura.

- Relación arco-nodo.

Métricas Orientadas a Clase
Estas métricas se incluyen dentro de las Métricas de Diseño Arquitectónico y según los autores Chidamber y Kemerer han propuesto seis métricas basadas en clases para sistemas OO:

- Métodos ponderados por clase (MPC).
- Árbol de profundidad de herencia (APH).
- Número de descendiente (NDD).
- Acoplamiento entre clases objeto (ACO).
- Respuesta para una clase (RPC).
- Carencia de cohesión en los métodos (CCM).

En su libro sobre métricas OO, Lorenz y Kidd separan las métricas basadas en clases en cuatro amplias categorías: tamaño, herencia, valores internos y valores externos. Las métricas orientadas al tamaño para las clases OO se centran en el recuento de atributos y operaciones para cada clase individual, y los valores promedio para el sistema OO como un todo. Las métricas basadas en la herencia se centran en la forma en que las operaciones se reutilizan en la jerarquía de clases. Las métricas para valores internos de clase examinan la cohesión y los aspectos orientados al código; las métricas orientadas a valores externos, examinan el acoplamiento y la reutilización. [23].

- Tamaño de clase (TC).
- Número de operaciones redefinidas por una subclase (NOR).
- Número de operaciones añadidas por una subclase (NOA).

Métricas de Diseño a Nivel de Componentes
Se incluyen dentro de las Métricas del Modelo de Diseño y se centran en las características internas de los componentes del software e incluyen las medidas de las “3C”: Cohesión (Cohesión); Coupling (Acoplamiento) y Complexity (Complejidad). Pueden aplicarse antes o después de tener el código. [12]

Métricas de Cohesión
Son Métricas de Diseño a Nivel de Componentes y se encargan de medir si la cohesión se puede usar en el trabajo de Bieman y Ott, el cual define varios conceptos:

- número de elementos (tokens),
- rebanada de datos (data slice),
- adhesivo (glue) y superadhesivo (superglue).

Con ellas se calcula:
Cohesión funcional fuerte: CFF = número de superadhesivos / número de elementos.
Cohesión Funcional Débil: CFD = número de adhesivos / número de elementos.
Mientras más cercano sean CFF ó CFD a 1, mayor será la cohesión del módulo. [12]

Métricas de Acoplamiento

Son Métricas de Diseño a Nivel de Componentes y proporcionan una indicación de la conectividad de un módulo con otros propuestos por Dhama en 1995. Constituyen medidas para el acoplamiento de flujo de datos de control. Incluye algunos parámetros a tener en cuenta:

- d_i = número de parámetros de datos de entrada.
- c_i = número de parámetros de control de entrada.
- d_o = número de parámetros de datos de salida.
- c_o = número de parámetros de control de salida.

Como medidas para el acoplamiento global esta métrica incluye el número de variables globales usadas como datos (g_d) y el número de variables globales usadas como control (g_c) y dentro de las medidas para el acoplamiento de entorno incluye el número de módulos llamados (w) y el número de módulos que llaman al módulo en cuestión (r). El indicador de acoplamiento de módulo: $m_c = k / M$, donde k = 1 y la constante de proporcionalidad es:

$M = d_i + a \times c_i + d_o + b \times c_o + g_d + c \times g_c + w + r$, con a = b = c = 2. Cuando mayor es m_c, menor es el acoplamiento del módulo. [12]

Métricas del Código Fuente

La forma de calcularlas es con las siguientes cantidades:

- n_1 = número de operadores diferentes en el programa.
- n_2 = número de operandos distintos en el programa.
- N_1 = número total de veces que aparecen los operadores.
- N_2 = número total de veces que aparecen los operandos.

Luego las métricas de Halstead, serían:

Longitud global del programa: $N = n_1 \log_2 n_1 + n_2 \log_2 n_2$.

Volumen del programa: $V = N \log_2 (n_1 + n_2)$.

Volumen compacto (ya que V varía dependiendo del lenguaje): $L = 2 / n_1 \times n_2 / N_2$. [12]

Métricas de Prueba

Las métricas de prueba que existen se concentran en el proceso de prueba y no en las características técnicas de la prueba. Los responsables de la prueba, se guían por las métricas de análisis, diseño y código.

- Puntos de Función.
- Bang.
- Halstead.
- Complejidad ciclomática. [12]

Métricas de Pruebas de Unidad

Se incluyen dentro de las Métricas de Prueba. El término prueba de unidad se refiere a la prueba individual de unidades separadas de un sistema de software. En sistemas orientados a objetos,

estas unidades son, típicamente, clases y métodos. Así, en el contexto que ocupa, prueba de unidad se refiere a la prueba individual de métodos y clases.
Las **pruebas de caja blanca** permiten examinar la estructura interna del programa realizando un seguimiento del código fuente según va ejecutando los casos de prueba, de manera que se determinan concretamente las instrucciones, bloques en los que existen errores. Para la realización de las pruebas de unidad se diseñan casos de prueba que se encargan de examinar la lógica del sistema. Los casos de prueba garantizan que se ejerciten todos los caminos independientes de cada módulo o clase y todas las decisiones lógicas así como que se ejecuten todos los bucles y las estructuras de datos internas.
Las **pruebas de caja negra** se llevan a cabo sobre la interfaz del software, y es completamente indiferente el comportamiento interno y la estructura del programa. Los casos de prueba de la caja negra pretenden demostrar que las funciones del sistema son operativas; que las entradas son aceptadas de forma adecuada y que las salidas correspondientes son las correctas. Además garantizan que la información externa se mantiene. Los errores esperados durante la realización de las pruebas de caja negra al sistema informático que se propone se agrupan en las siguientes categorías:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

Las pruebas de regresión por su parte son una estrategia de prueba en la cual las pruebas que se han ejecutado anteriormente se vuelven a realizar en la nueva versión modificada, para asegurar la calidad después de añadir la nueva funcionalidad. El propósito de estas pruebas es asegurar que los defectos identificados en la ejecución anterior de la prueba se han corregido y que los cambios realizados no han introducido nuevos defectos o reintroducido defectos anteriores.

1.4 Conclusiones

Como resultado de la investigación y el análisis bibliográfico realizado, a lo largo del capítulo han sido estudiados los diferentes puntos que están relacionados con la investigación. Las herramientas, frameworks y metodologías a utilizar han sido objeto de análisis, mostrándose sus características y su funcionamiento, se han seleccionado los diferentes factores que intervendrán en el desarrollo del software propuesto por lo que queda vencido el objetivo de este capítulo.

CAPÍTULO 2: ARQUITECTURA, DISEÑO E IMPLEMENTACIÓN

2.1 **Introducción**.

En este capítulo se tratarán los temas aplicados al sistema propuesto como es la arquitectura del sistema y el estilo arquitectónico usado, se abordará el tema de los estándares de codificación debido a la importancia que representa para todo el desarrollo de un proyecto. También en este capítulo se hablará de los patrones de diseño y los patrones de asignación de responsabilidad implementados por el framework propuesto para la solución, estarán los diferentes diagramas de diseño e implementación del sistema como son: los diagramas de clases del diseño, diagramas de iteración del diseño, el diagrama de componentes y de despliegue. Finalmente las conclusiones del capítulo.

2.2 **Arquitectura del Sistema**.

La palabra arquitectura proviene del griego “αρχ”, cuyo significado es “jefe, quien tiene el mando”, y de “τεκτων” que significa “constructor o carpintero”. Así, para los antiguos griegos el arquitecto es el jefe o el capataz de la construcción y la arquitectura es la técnica o el arte de quien realiza el proyecto.
Según Clements gran conocedor del tema, la arquitectura de software es, a grandes rasgos, una vista del sistema en la que se incluyen los componentes principales del mismo, la conducta de esos componentes y las formas en que los componentes interactúan en perfecta coordinación para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando un alto nivel de comprensión y la supresión del detalle inherente a gran parte de las abstracciones. [36]

La IEEE Std 1471-2000 plantea otra definición de arquitectura que se ha tomado como oficial por todas las instituciones que desarrollan software a nivel mundial, como ha sido por ejemplo Microsoft. La definición de IEEE Std 1471-2000 parte de la idea de que: “La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución”. [35]

El subsistema de Gestión de Cuadro y Personal de Apoyo consta de seis módulos que ayudarán a un mejor desarrollo del sistema propuesto, estos módulos son:

- Gestión de Cuadro.
- Personal de Apoyo.
- Capacitación.
- Captación de Plantillas.
- Generalidades.
- Reportes y Búsqueda.

La arquitectura para el subsistema de Gestión de Cuadro y Personal de Apoyo (GCPA) representa la base para todo el funcionamiento de la aplicación, es el pilar principal del producto que se quiere

construir. El sistema se desarrollará en plataforma Linux (Debian), con Apache, como gestor de base de datos Postgres y como lenguaje de programación se seleccionó al PHP5. El principal motivo de esta decisión es la política de migración hacia software libre que tiene la Fiscalía General de la República por las grandes ventajas y la soberanía tecnológica que esto genera. El sistema que se construirá será una aplicación Web debido a que este es un requerimiento del cliente y además desarrollar aplicaciones Web brinda las siguientes ventajas:

- Agilidad y sencillez en el despliegue y actualización de la aplicación.
- Disminuye los requerimientos de hardware en las PC clientes y por tanto el costo de inversión.
- Ganancia en la seguridad de la aplicación al estar localizada en un servidor alejado del acceso directo del usuario.
- Amplio soporte de frameworks que apoyan el desarrollo en la plataforma.
- Mayor experiencia del equipo de desarrollo en estas aplicaciones.

2.2.1 **Estilo Arquitectónico Aplicado.**

El Sistema de Gestión de Cuadro y Personal de Apoyo (GCPA) en su mayoría se enmarca dentro de los sistemas de gestión, debido a que cuenta con una base de datos donde la información es gestionada según las necesidades de los usuarios, el estilo arquitectónico seleccionado después del estudio realizado en el capítulo anterior es el de Arquitectura en Capas por las grandes ventajas que este presenta, las capas que componen el sistema son:

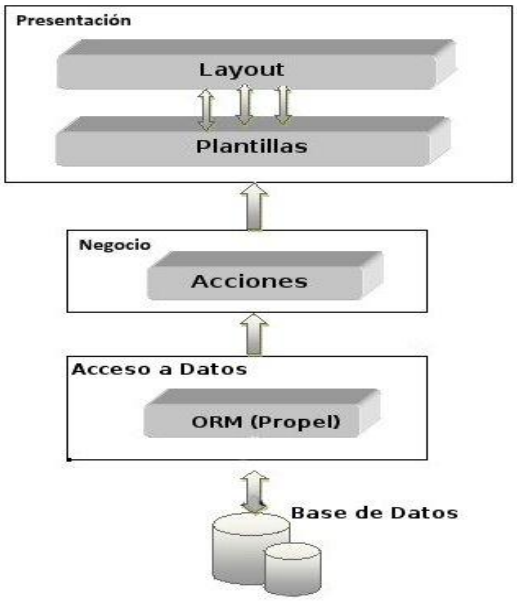

Figura: 2.2.1.1 Estilo en Capas.

Presentación (Vista):
Es la capa que ve el usuario (algunos la denominan "capa de usuario"), presenta el sistema al usuario, le comunica y captura la información del usuario dando un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio.

Negocio (Controlador):
En esta capa se almacenarán todos los componentes que gestionarán el negocio de la aplicación.

Datos (Modelo):
Esta capa estará dividida en dos capas, la capa Acceso a Datos y la capa Datos, en la capa acceso a datos se almacenarán todas las clases que mapean los objetos de la base de datos. En la capa Datos se encontrará la base de datos con sus tablas, relaciones y funciones.

2.4 Estándar de Codificación.

Al comenzar un proyecto de software, es necesario establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada, es decir que estos hablen en el mismo lenguaje. Cuando el proyecto de software incorpore código fuente previo, o bien cuando realice el mantenimiento de un sistema de software creado anteriormente, el estándar de codificación establece cómo operar con la base de código existente.
En el proyecto Sistema de Gestión Fiscal el estándar de codificación constituye el alma del proyecto debido a que se trabaja con los estándares que trae el Symfony para lograr su completa comprensión, más los estándares que exigen los lenguajes de PHP y HTML que ayudan a una correcta comprensión de todo el código por parte de los desarrolladores.
Para que exista uniformidad en el código escrito y para que los futuros desarrolladores entiendan el mismo, debe utilizarse el estilo Camello con algunas especificaciones las cuales se muestran a continuación:

Indentación
Las llaves ({**)** de inicio deben estar al principio y al cierre (}**)** debajo de la declaración a la que pertenecen, ejemplo:

```
public function executeIndex()
    {
      return $this->forward('registrarCuadro', 'list');
    }
```

Líneas en blanco
Se debe colocar una línea en blanco antes y después de la declaración de una estructura, una clase o una función.

```
public function executeIndex()
  {
       return $this->forward('registrarCuadro', 'list');
  }

public function executeCreate()
  {
       $this->tbcuadro = new Tbcuadro();
       $this->tbpersona = new Tbpersona();
       $this->setTemplate('edit');
  }
```

Regla para identificadores

En el caso de variables y atributos en minúscula y los nombres de los métodos deben comenzar con mayúscula y en caso de que sea una palabra compuesta la primera letra de la que empieza a continuación de la anterior debe ser mayúscula, ejemplo:

```
public function executeList()
   {
       $this->tbcuadros = TbcuadroPeer::doSelect(new Criteria());
   }
```

Comentarios y Descripciones

Para el caso de las funciones se debe poner una breve descripción de la misma así como el tipo de parámetro que recibe y devuelve, ejemplo:

```
/*
* Dado el identificador de una Persona pasada desde la  vista,
* realiza una consulta a la base de datos y devuelve los datos
* De la persona para que sean editados.
* Paramametro $idpersona
* return      Cuadro
*/
 public function executeEdit()
   {
      $this->tbcuadro = TbcuadroPeer::retrieveByPk($this- >getRequestParameter('idpersona'));

      $this->forward404Unless($this->tbcuadro);

   }
```

Otra forma de hacer un comentario para especificar que función realiza un fragmento de código es la siguiente:

```
    //Para guardar la imagen
```

```
$imagen = new Nimagen();
$fileName = $this->getRequest()->getFileName('file');
$this->getRequest()->moveFile('file',sfConfig::get('sf_upload_dir_name').'/'.$fileName);
$imagen->setImagen($fileName);

$imagen->save();
```

2.4 Patrones

2.4.1 Patrones de Diseño Empleados.

Symfony se ha construido teniendo en cuenta muchos de los patrones de diseño comunes como son singleton, abstract factory, decorator, a continuación algunos patrones que contiene este framework.

En la categoría Creacionales:

Singleton (Instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia desde cualquier parte del framework.

Abstract Factory (Fábrica abstracta): Permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando. Cuando el framework necesita por ejemplo crear un nuevo objeto para una petición, busca en la definición de la factoría el nombre de la clase que se debe utilizar para esta tarea.

Factoría:

sfContext::getInstance()->getI18N()
sfContext::getInstance()->getRouting()
sfContext::getInstance()->getLogger()

En la categoría Estructurales:

Decorator (Envoltorio): Añade funcionalidad a una clase, dinámicamente. El archivo layout.php, que también se denomina plantilla global, almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el layout, o si se mira desde el otro punto de vista, el layout decora la plantilla. Este comportamiento es una implementación del patrón de diseño.
apps/frontend/templates/layout.php

2.4.2 Patrones GRASP Empleados

Entre los patrones de asignación de responsabilidad que presenta Symfony están:

Creador

En las clases Actions se encuentran las acciones definidas para el Sistema de Gestión Fiscal y se

ejecutan cada una de ellas. En las acciones se crean los objetos de las clases que representan las entidades, evidenciando de este modo que la clase Actions es "creador" de dichas entidades.

Experto

Este es uno de los más utilizados, puesto que Propel es la librería externa que utiliza Synfony para realizar su capa de abstracción en el modelo, encapsula toda la lógica de los datos y son generadas las clases con todas las funcionalidades comunes de las entidades.

Alta Cohesión

Symfony permite asignar responsabilidades con una alta cohesión, por ejemplo la clase Actions tiene la responsabilidad de definir las acciones para las plantillas y colabora con otras para realizar diferentes operaciones, instanciar objetos y acceder a las properties, es decir, está formada por diferentes funcionalidades que se encuentran estrechamente relacionadas proporcionando que el software sea flexible frente a grandes cambios.

Controlador

Todas las peticiones Web son manejadas por un solo controlador frontal (sfActions), que es el punto de entrada único de toda la aplicación en un entorno determinado. Cuando el controlador frontal recibe una petición utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL entrada por el usuario.

sfController es la clase del controlador. Se encarga de decodificar la petición y transferirla a la acción correspondiente.

Bajo Acoplamiento

Las clases Actions heredan solamente de sfActions para lograr un bajo acoplamiento de clases.

2.5 Diseño e Implementación.

En este epígrafe se reflejaran los aspectos del diseño y la implementación partiendo de los diagramas de secuencia y de clases del diseño, el diagrama de componentes y el modelo de despliegue.

2.5.1 Diagramas de Interacciones del Diseño.

Los diagramas de secuencia son acciones que comienzan cuando un caso de uso es invocado por un actor mediante el envío de mensajes, lo cual permite que comience el flujo de datos entre los distintos objetos que intervienen en el funcionamiento del sistema, estos diagramas ayudan a comprender mejor el paso de la información en el sistema mediante los mensajes que se envían entre objetos de forma detallada y ordenada en el tiempo.

A continuación se muestran los diagramas de secuencia de los 6 módulos en los que se divide el Subsistema de Gestión de Cuadro y Personal de Apoyo.

Gestión de Cuadro

Los diagramas de secuencia del módulo de gestión de cuadro se dividieron en los casos de usos que se muestran a continuación.

- Registrar Datos Personales del Cuadro.
- Registrar Datos Laborales del Cuadro.
- Registrar Datos de la Evaluación del Cuadro.
- Registrar Acuerdo del Consejo de Estado para Cese de Funciones.
- Registrar Acuerdo del Consejo de Estado.
- Registrar Acuerdo de la Asamblea Nacional del Poder Popular.
- Registrar Acuerdo de la Asamblea Nacional del Poder Popular para Cese de Funciones.
- Búsqueda Simple.
- Registrar Resolución de Movimiento.
- Registrar Resolución de Designación y Nombramiento.
- Registrar Resolución de Cese de las Funciones.
- Registrar Permisos.
- Modificar Permisos.
- Modificar Datos Personales del Cuadro.
- Modificar Datos Laborales del Cuadro.
- Modificar Datos de la Evaluación del Cuadro.

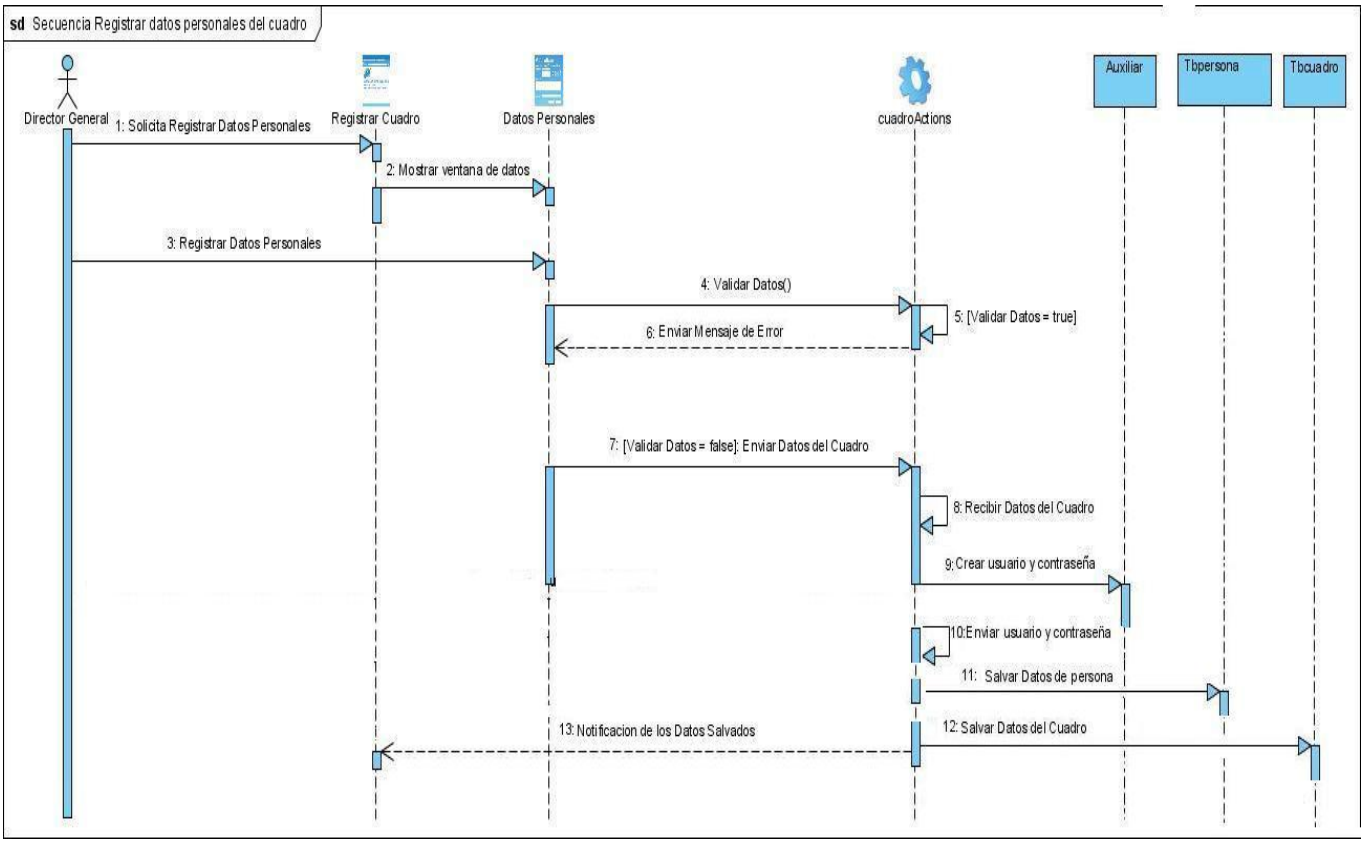

Figura 2.6.1.1 Diagrama de Secuencia: Registrar Datos Personales del Cuadro.

En este caso de uso se introducen los datos personales de los cuadros para la creación del expediente de los mismos. Luego de introducir los datos del cuadro se guarda la información.
Para más información de los diagramas presentados en este módulo (Ver Anexo1).

Personal de Apoyo

Los diagramas de secuencia del módulo de personal de apoyo se dividieron en los casos de usos que se muestran a continuación.

- Registrar Datos de Captación del Asistente Fiscal.
- Registrar Permisos del Personal de Apoyo.
- Registrar Datos de Captación de Otras Personas.
- Modificar Permisos del Personal de Apoyo.
- Modificar Datos de Captación del Asistente Fiscal.
- Modificar Datos de Captación de Otras Personas.
- Adicionar Asignaturas Vencidas.
- Visualizar Captación de Asistente Fiscal.
- Visualizar Captación de Otras Personas.

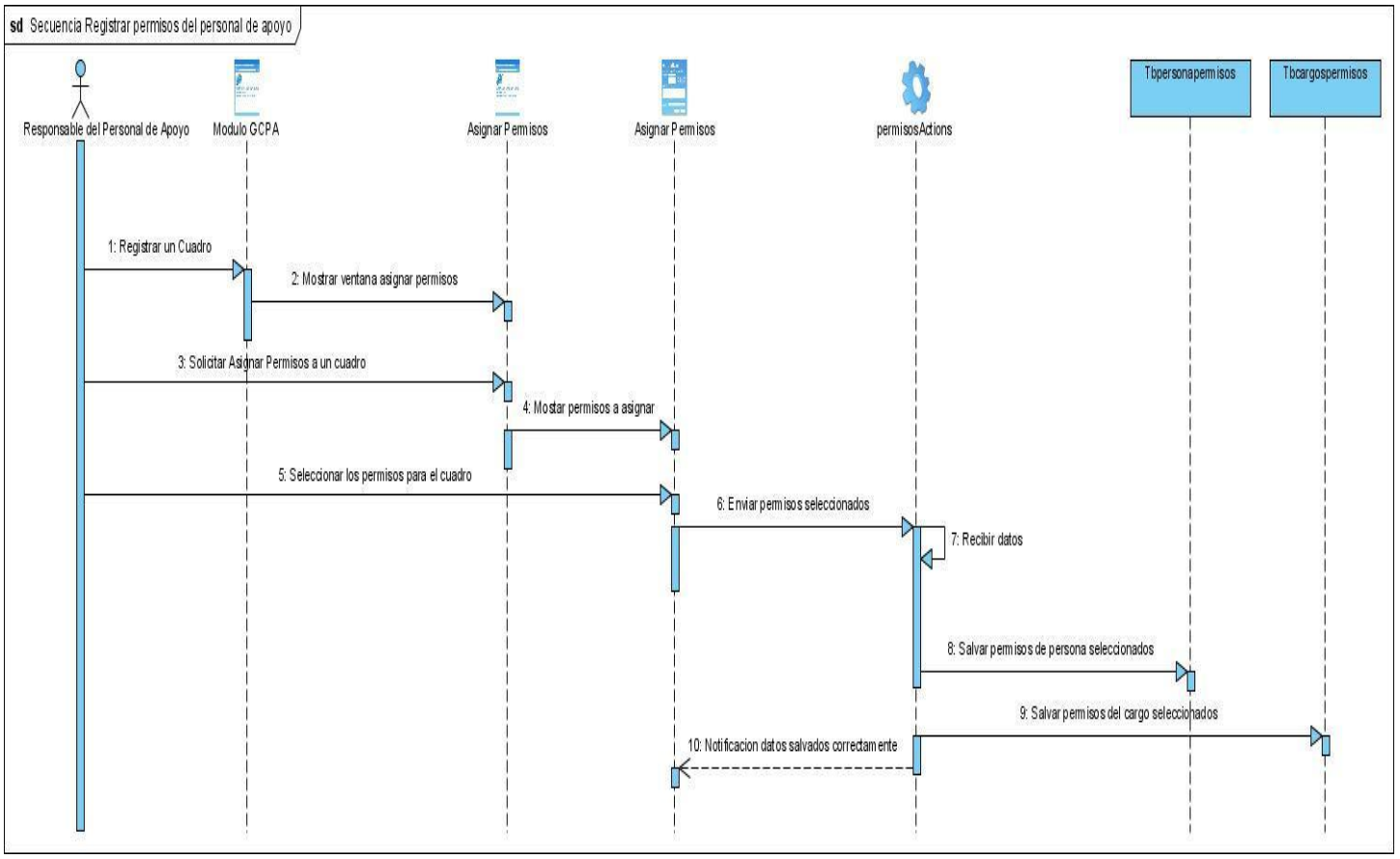

Figura 2.6.1.2 Diagrama de Secuencia: Registrar Permisos del Personal de Apoyo.

Este caso de uso permite al responsable de personal de apoyo del sistema registrar los permisos de los usuarios en el sistema.
Para más información de los diagramas presentados en este módulo (Ver Anexo2).

Capacitación
Los diagramas de secuencia del módulo de capacitación se dividieron en los casos de usos que se muestran a continuación.

- Registrar Capacitación del Cuadro.
- Registrar Categoría Docente y Fiscal del Cuadro.
- Modificar Datos de Capacitación.
- Modificar Categoría Docente y Fiscal.
- Visualizar Categoría Docente y Fiscal.
- Visualizar Datos de Capacitación.

Para más información de los diagramas presentados en este módulo (Ver Anexo3).

Captación de Planillas.
Los diagramas de secuencia del módulo de captación de planillas se dividieron en los casos de usos que se muestran a continuación.

- Registrar Datos de Captación de Plantilla.
- Modificar Datos de Captación de Plantilla.

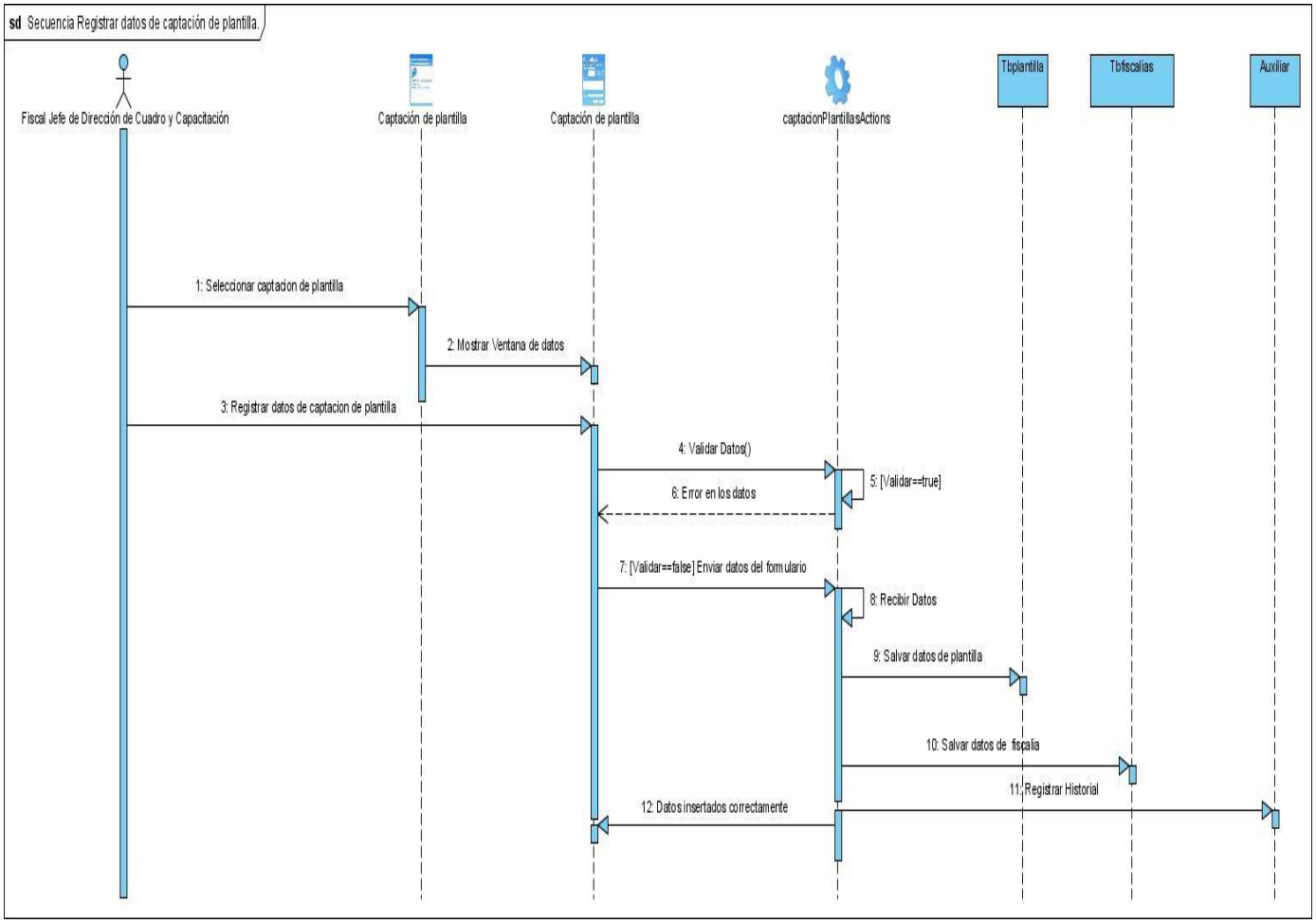

Figura 2.6.1.4 Diagrama de Secuencia: Registrar Datos de Captación de Plantilla.

En este caso de uso se registran los datos relacionados con la captación de plantillas.
Para más información de los diagramas presentados en este módulo (Ver Anexo4).

Generalidades.
Los diagramas de secuencia del módulo de generalidades se dividieron en los casos de usos que se muestran a continuación.

- Autenticar Usuario.

- Cambiar Contraseña.
- Mostrar Perfil.
- Olvidar Contraseña.
- Visualizar Historial.
- Visualizar Notificaciones.

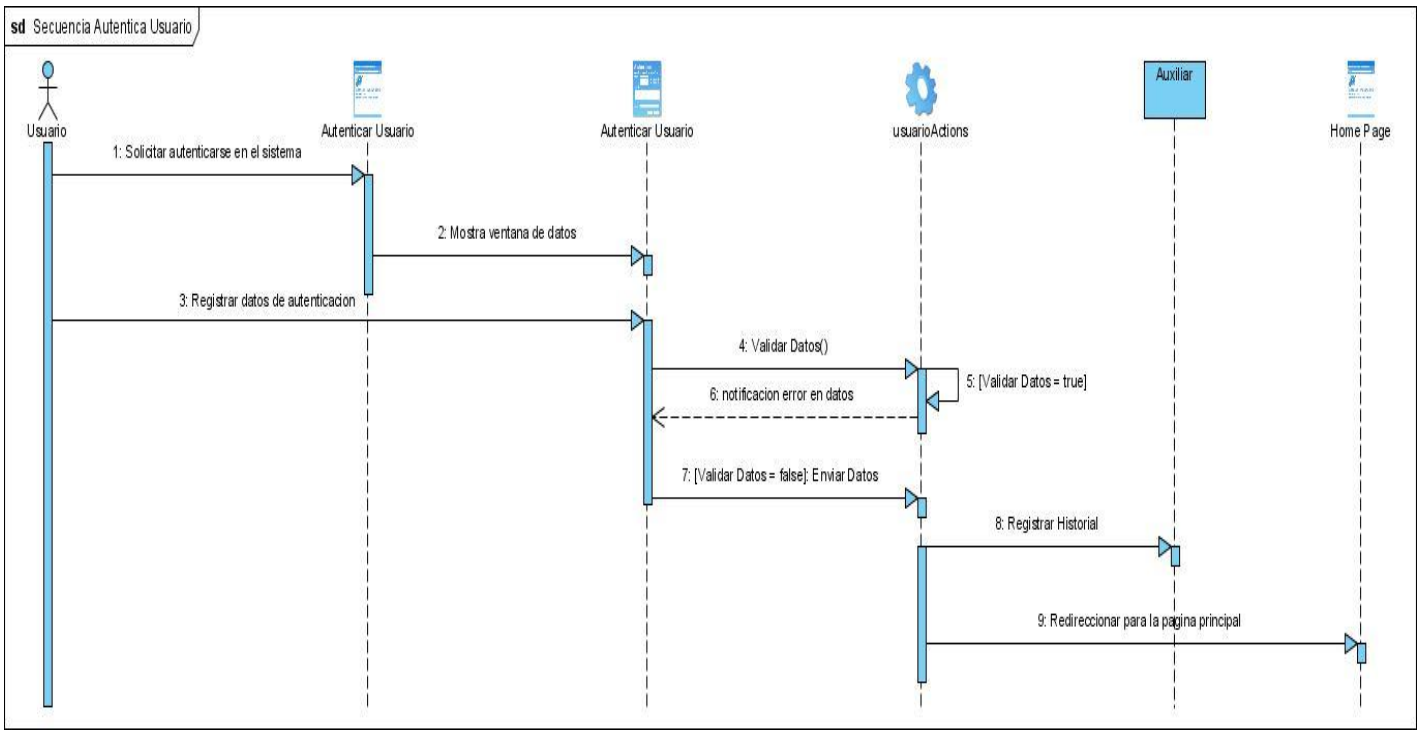

Figura 2.6.1.5 Diagrama de Secuencia: Autenticar Usuario.

Este caso de uso comienza cuando un usuario accede al sistema, y debe autenticarse, poner su nombre de usuario y contraseña, si son correctos los mismos tiene acceso, sino el sistema muestra algún mensaje. Terminando de esta forma el caso de uso.
Para más información de los diagramas presentados en este módulo (Ver Anexo5).

Reporte y Búsqueda
Los diagramas de secuencia del módulo de reporte y búsqueda se dividieron en los casos de usos que se muestran a continuación.

- Búsqueda Avanzada de Capacitación.
- Búsqueda Avanzada de Cuadro.
- Categoría Docente, Categoría fiscal y Unión Nacional de Juristas de Cuba.
- Generar Currículo del Cuadro.
- Generar Expediente del Cuadro.

- Generar Expediente del Personal de Apoyo.
- Mostrar Componente de Preparación y Superación.
- Mostrar Dirigentes por Etnia y Militancia.
- Mostrar Etnia por Provincias.
- Mostrar Evaluaciones.
- Mostrar Experiencia Fiscal.
- Mostrar Fiscales de la Reserva por Provincias.
- Mostrar Militantes por Provincias.
- Mostrar Motivos de Bajas por Provincias.
- Mostrar Nivel de Dominio de Idiomas de los Fiscales.
- Mostrar Promedio de Edad por Provincias, Municipio y Cargos.

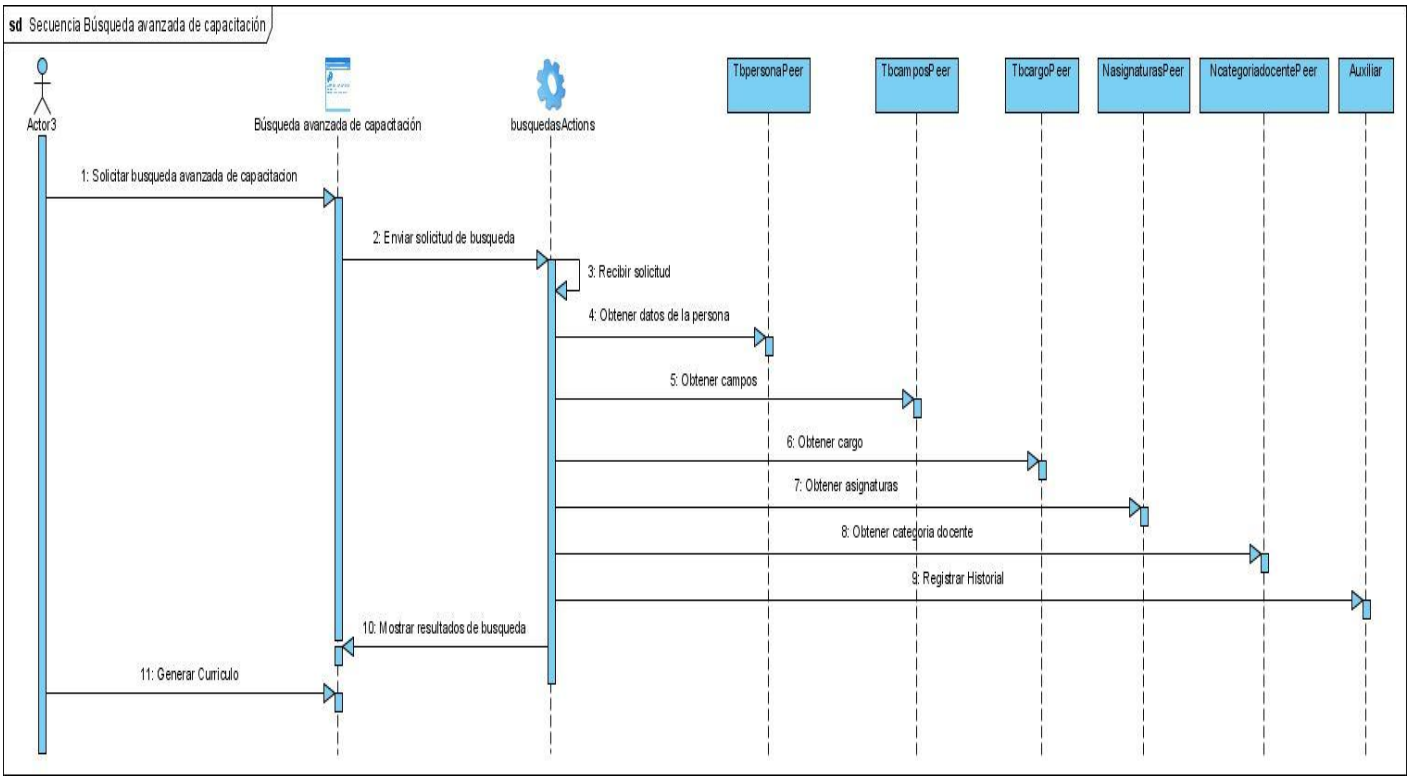

Figura 2.6.1.6 Diagrama de Secuencia: Búsqueda Avanzada de Capacitación.

En este caso de uso se pueden realizar las búsquedas avanzadas relacionadas con los datos de la capacitación de los fiscales que se encuentran registrados en el sistema. Se seleccionan cuales son los datos que quiero buscar y se acepta la acción, luego el sistema muestra un listado con los nombres de las personas que coinciden con la búsqueda realizada y da la posibilidad de acceder al currículo de los mismos.
Para más información de los diagramas presentados en este módulo (Ver Anexo6).

2.5.2 Diagramas de Clases del Diseño.

Los diagramas de clases del diseño muestran una representación gráfica de las clases que contiene un software y sus especificaciones, definen las características de cada clase, métodos, interfaces, colaboraciones, relaciones, dependencias y generalizaciones de forma tal que representa la vista estática del sistema.
A continuación se muestran los diagramas de clases del diseño por módulos del Subsistema de Gestión de Cuadro y Personal de Apoyo.

Gestión de Cuadro
Los diagramas de clases del diseño del módulo gestión de cuadro se dividieron en los casos de usos mencionados en los diagramas de secuencia.

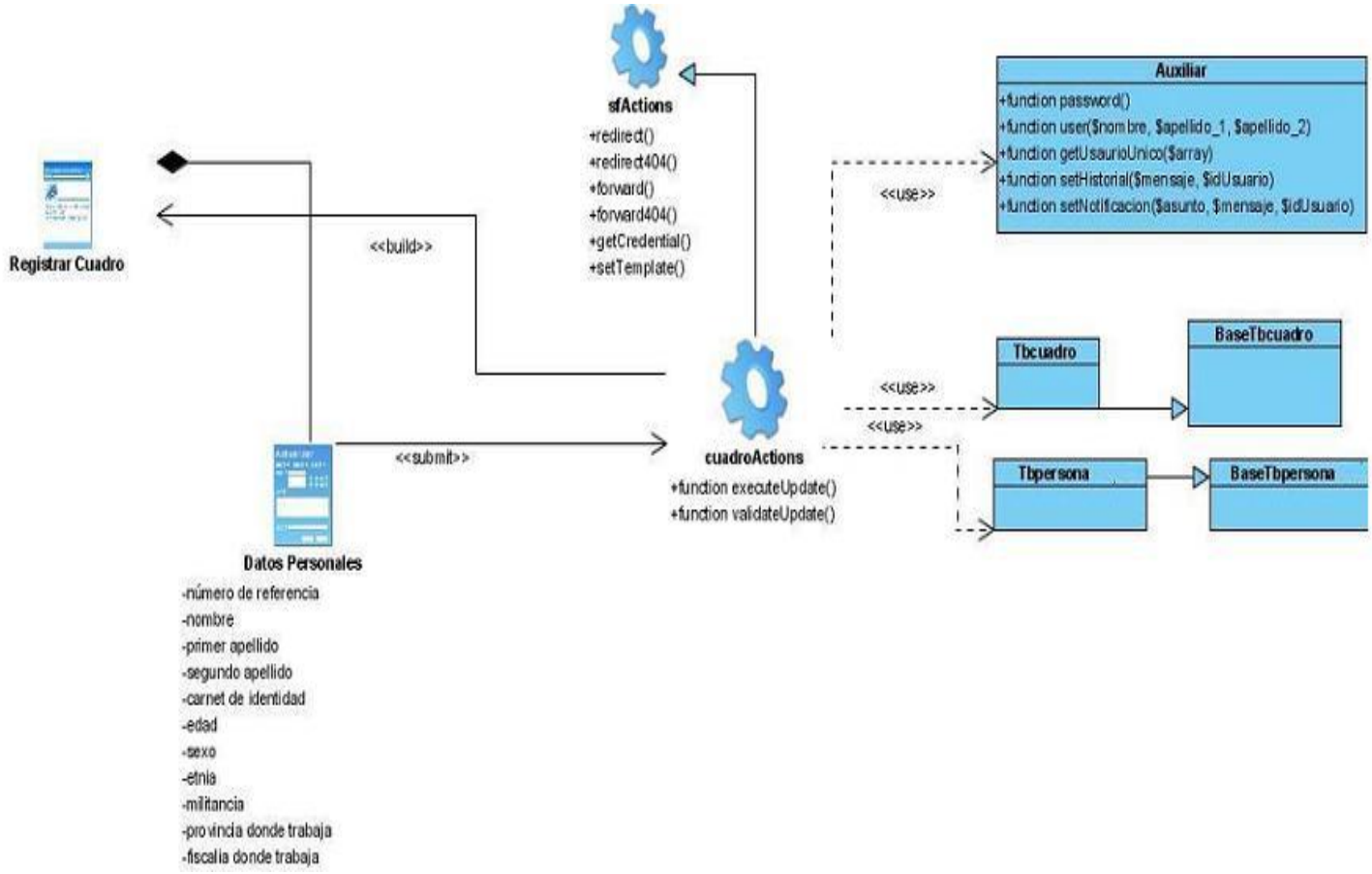

Figura 2.6.2.1 Diagrama de Clase del Diseño: Registrar Datos Personales del Cuadro.

Para más información de los diagramas de clases presentados en este módulo (Ver Anexo7).

Personal de Apoyo

Los diagramas de clases del diseño del módulo personal de apoyo se dividieron en los casos de usos mencionados en los diagramas de secuencia.

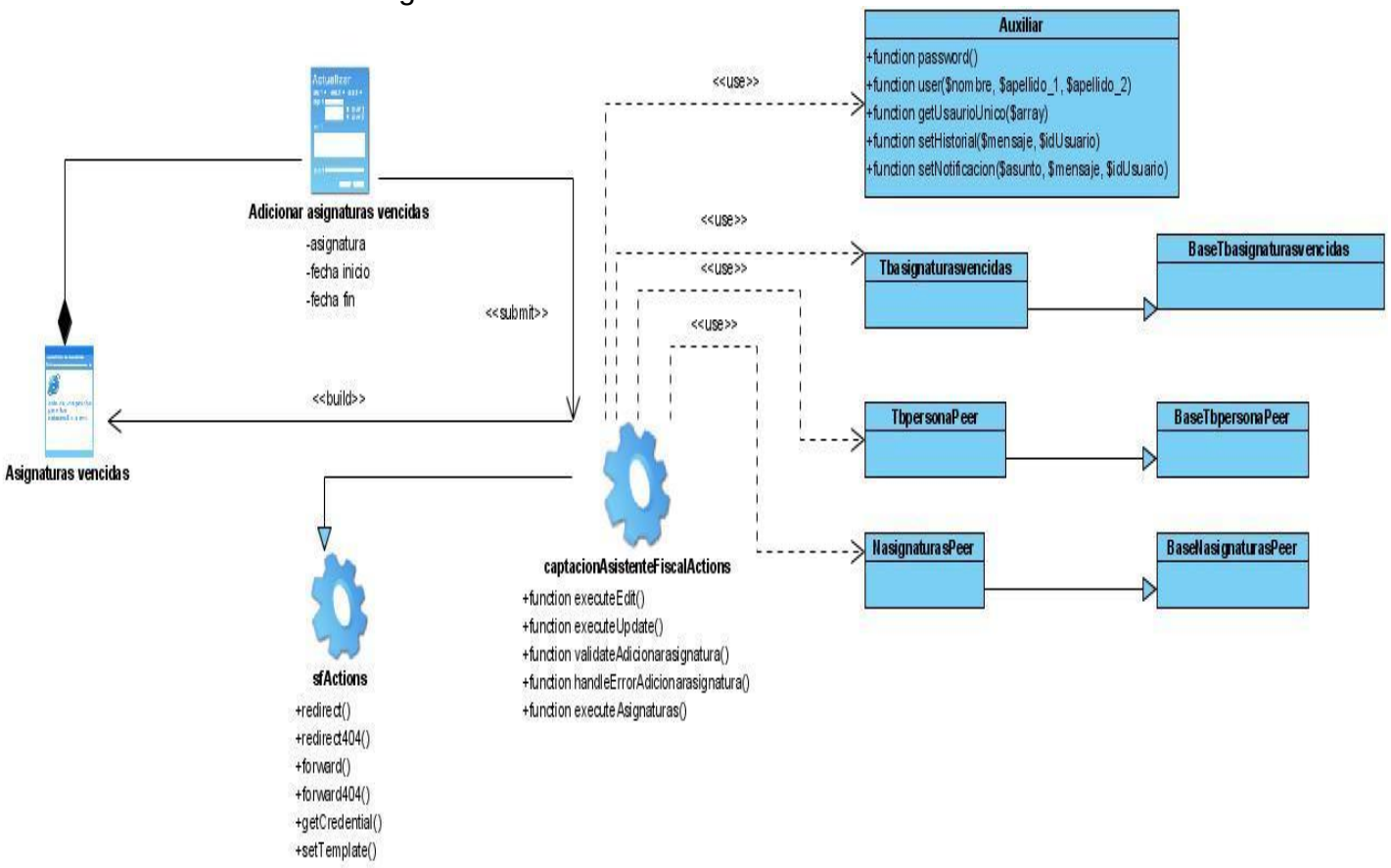

Figura 2.6.2.2 Diagrama de Clase del Diseño: Adicionar Asignaturas Vencidas.

Para más información de los diagramas de clases presentados en este módulo (Ver Anexo8).

Capacitación

Los diagramas de clases del diseño del módulo de capacitación se dividieron en los casos de usos mencionados en los diagramas de secuencia.

Para más información de los diagramas de clases presentados en este módulo (Ver Anexo9).

Captación de Planillas.

Los diagramas de clases del diseño del módulo de captación de plantilla se dividieron en los casos de usos mencionados en los diagramas de secuencia.

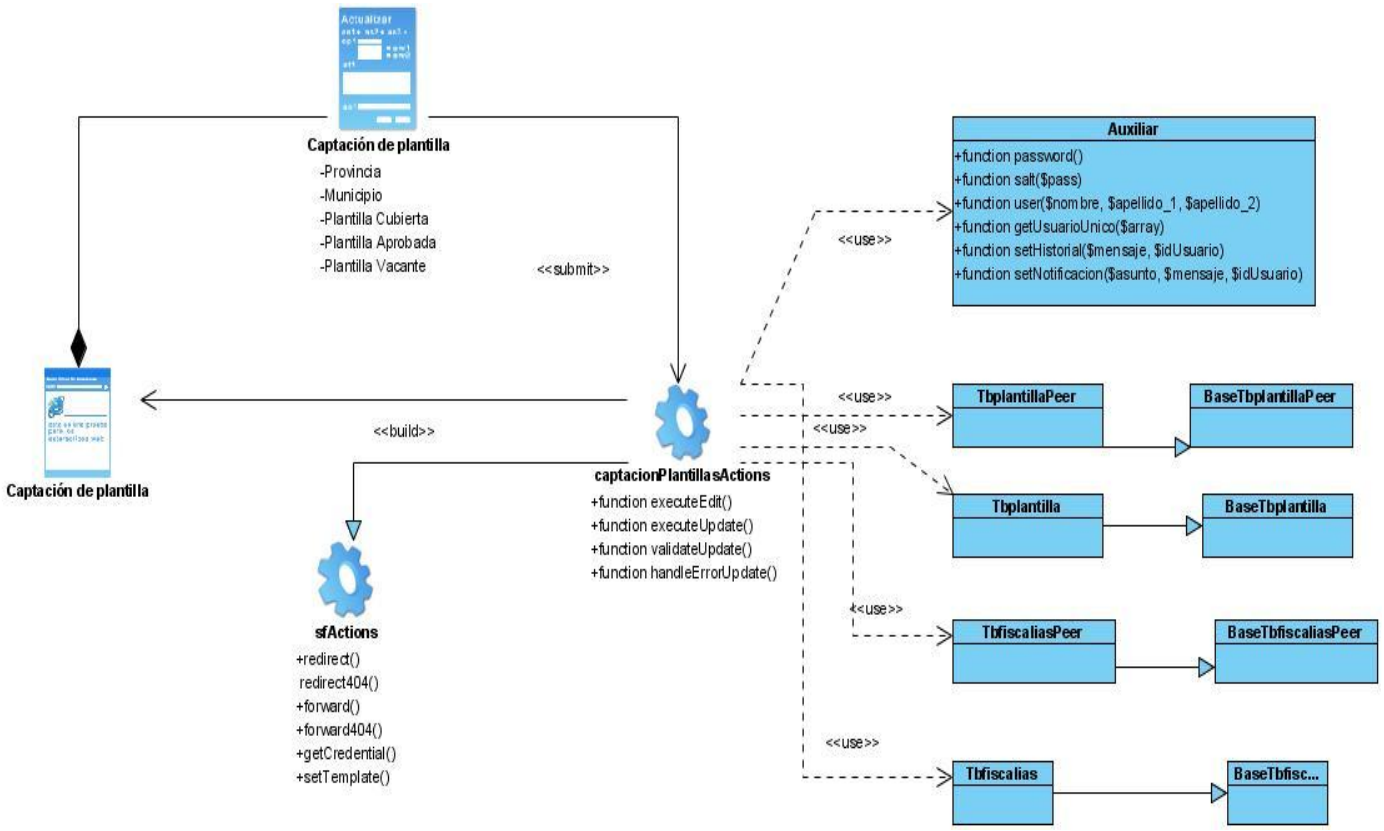

Figura 2.6.2.4 Diagrama de Clase del Diseño: Registrar Datos de Captación de Plantilla.

Para más información de los diagramas de clases presentados en este módulo (Ver Anexo10).

Generalidades.

Los diagramas de clases del diseño del módulo de generalidades se dividieron en los casos de usos mencionados en los diagramas de secuencia.

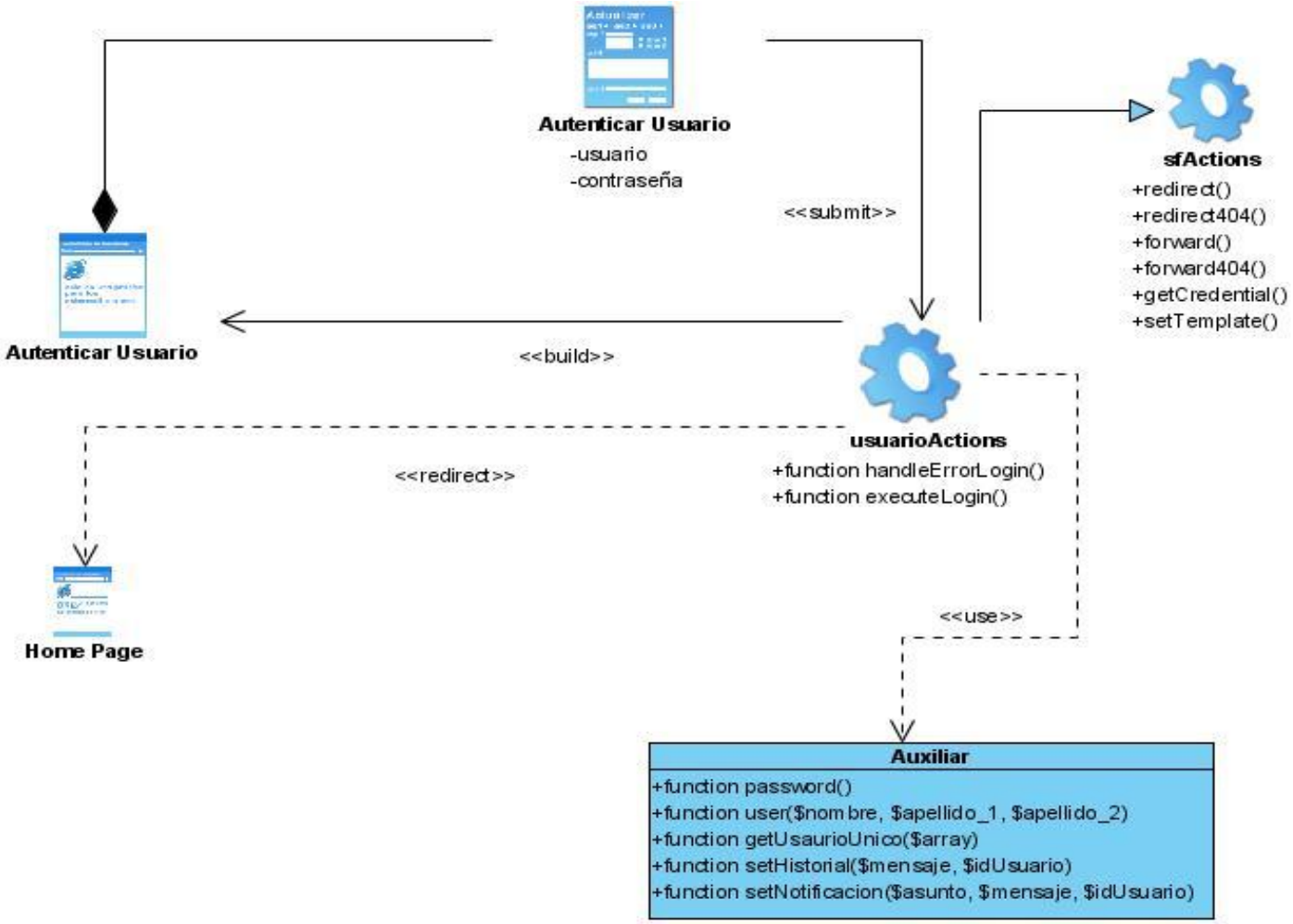

Figura 2.6.2.5 Diagrama de Clase del Diseño: Autenticar Usuario.

Para más información de los diagramas de clases presentados en este módulo (Ver Anexo11).

Reportes y Búsqueda

Los diagramas de clases del diseño del módulo de reportes y búsqueda se dividieron en los casos de usos mencionados en los diagramas de secuencia.

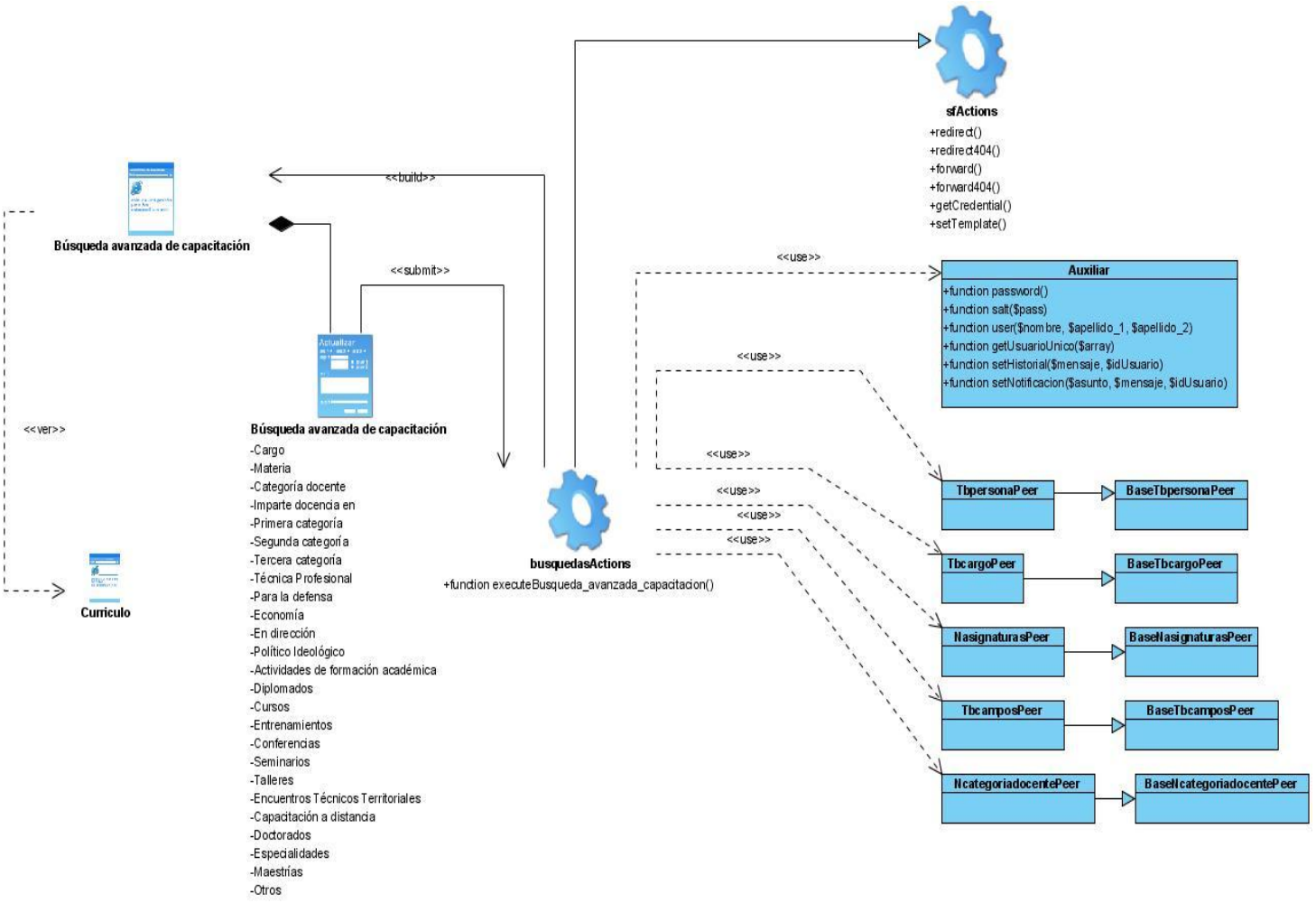

Figura 2.6.2.6 Diagrama de Clase del Diseño: Búsqueda Avanzada de Capacitación.

Para más información de los diagramas de clases presentados en este módulo (Ver Anexo12).

2.5.3 Modelo de Implementación.

El Modelo de implementación constituye un paso muy importante para el Proceso de Desarrollo de Software ya que es donde se planifica la integración del sistema en pequeños pasos manejables que permiten implementar clases, subsistemas y los componentes encontrados durante el diseño, estos modelos asocian componentes ejecutables al Diagrama de Despliegue y se prueban individualmente estos componentes para luego ser integrados al sistema.
En este modelo se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos. Este artefacto describe las dependencias entre los componentes, congregándolos en subsistemas organizados en capas y jerarquías.

El modelo de implementación está conformado por los Diagramas de Despliegue y el Diagramas de componentes, que son artefactos generados durante el flujo de trabajo de implementación.
Los Diagramas de Componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación mostrando las relaciones entre sus elementos. El uso más importante de estos diagramas es mostrar la estructura de alto nivel del modelo, especificando los subsistemas de implementación y sus dependencias.
A continuación se muestra el Diagrama de Componentes para el sistema.

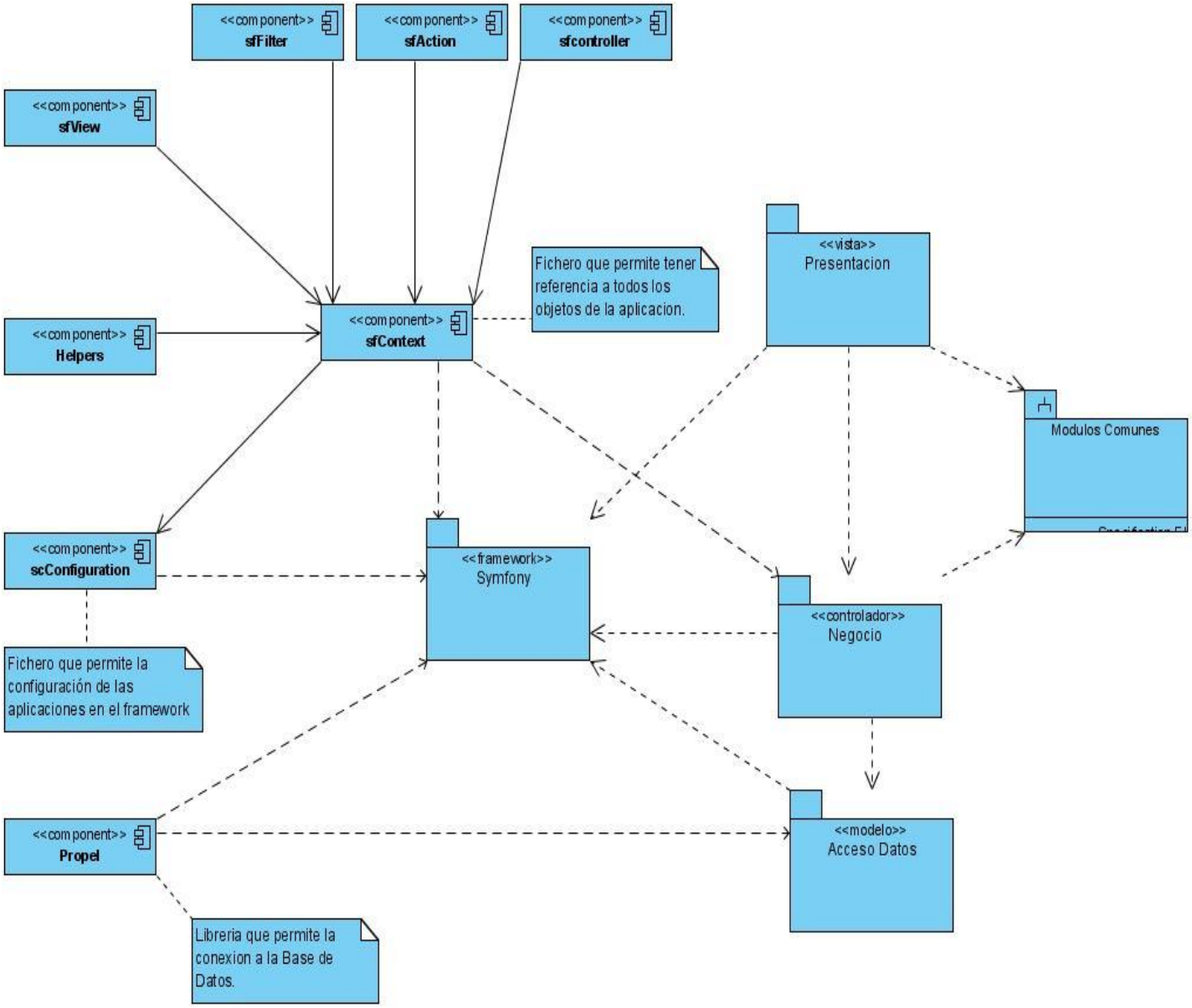

Figura 2.6.4.1 Diagrama de Componente de Symfony

2.5.4 Modelo de Despliegue.

El Modelo Físico/de Despliegue provee un modelo detallado de la forma en la que los componentes se desplegarán a lo largo de la infraestructura del sistema. Detalla las capacidades de red, las especificaciones del servidor, los requisitos de hardware y otra información relacionada al despliegue del sistema propuesto.
A continuación se muestra el modelo de despliegue del proyecto Sistema de Gestión Fiscal.

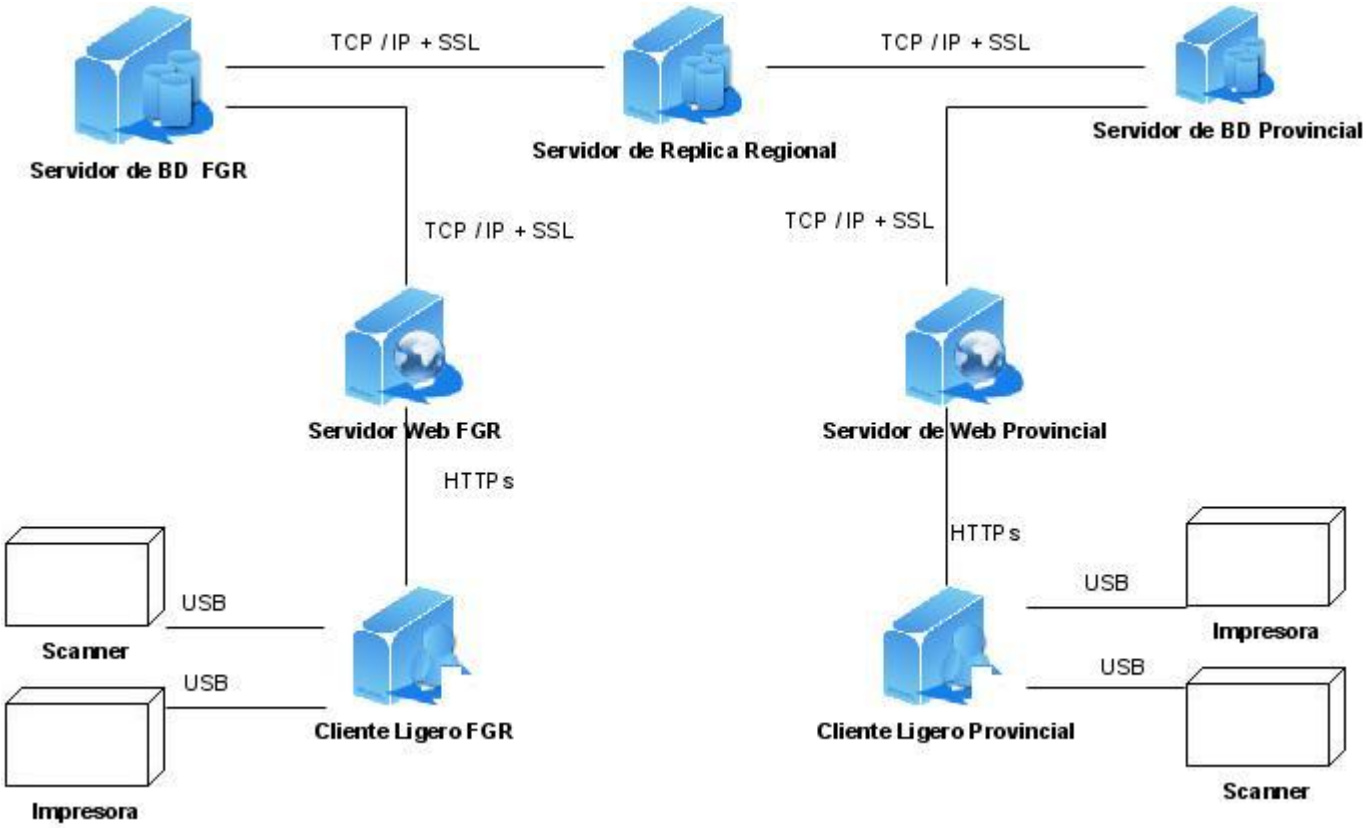

Figura 2.6.5.1 Modelo de Despliegue

2.7 Conclusiones.

A lo largo del presente capítulo se mostraron los resultados de la etapa de diseño e implementación del sistema. Se trataron temas como la arquitectura del sistema, el estilo arquitectónico aplicado, los estándares de codificación, los patrones de diseño y los patrones de asignación de responsabilidad empleados, los diagramas de interacciones y de clases del diseño, el diagrama de componentes del sistema y el modelo de despliegue de la solución. Estos resultados dan cumplimiento a los objetivos del capítulo, quedando listo el terreno para aplicarles las métricas al diseño realizado y las pruebas de caja negra al sistema propuesto.

CAPÍTULO 3: ANÁLISIS DE RESULTADOS

3.1 Introducción.

En este capítulo se tratarán los temas relacionados con las métricas aplicadas al sistema para lograr obtener un producto robusto y flexible que garantice mayor calidad en los proceso fiscales, también se realizarán las pruebas de Caja Negra para verificar que el sistema está en óptimas condiciones para su funcionamiento.

3.2 Métricas Aplicadas.

Con el objetivo de determinar el grado de calidad y fiabilidad del diseño que se propone se establecieron algunas métricas de diseño basadas en clases para medir categorías tales como tamaño, herencia, nivel de profundidad, ya que son aspectos orientados al código, a la cohesión, al acoplamiento y la reutilización. A continuación estarán algunas métricas aplicadas a la capa del negocio de la aplicación debido a que esta capa es una de las más importantes en todo el sistema.

3.2 .1 Métrica Tamaño de Clase (TC).

El tamaño general de una clase se puede determinar empleando métricas para saber el **número total de operaciones** (tanto operaciones heredadas como privadas de la instancia) que están encapsuladas dentro de la clase y encontrando el **número de atributos** (tanto atributos heredados como atributos privados de la Instancia) que están encapsulados en la clase.
Las medidas o umbrales para los parámetros de calidad han sido una polémica a nivel mundial en el diseño de sistemas. Algunos especialistas plantean umbrales para estas métricas según como se muestra a continuación, los cuales fueron aplicados al diseño.

Número de Operaciones y/o Atributos	
Métrica TC	**Umbral**
Pequeño	<=20
Medio	>20 y <=30
Grande	>30

Figura 3.2.1 Umbrales para la Métrica TC.

Los valores grandes de TC demuestran que una clase puede tener demasiada responsabilidad, lo cual reducirá la reutilizabilidad de la clase y complicará la implementación y la comprobación,

por otra parte cuanto menor sea el valor medio para el tamaño, más probable es que las clases existentes dentro del sistema se puedan reutilizar ampliamente.

Resultados.
Al aplicar la métrica TC en las clases presentes en la Capa del Negocio quedaron recogidos los siguientes datos:

Clases	No. Atributos	No. Operaciones
1. busquedasActions	0	12
2. capacitacionCuadroActions	0	13
3. captacionPlantillasActions	0	11
4. captacionAsistenteFiscalActions	0	16
5. captacionOtrasPersonasActions	0	13
6. reportesActions	0	29
7. resolucionActions	0	20
8. resolucionCeseActions	0	13
9. permisosActions	0	10
10. categoriaDocenteFiscalActions	0	13
11. acuerdoActions	0	22
12. acuerdoCeseActions	0	22
13. cuadroActions	0	26
14. Auxiliar	10	6
15. myUser	1	6

Tabla 1. Clases de la Capa de Negocio.

La mayoría de las clases que conforman esta capa están dentro de la categoría de pequeñas, lo que demuestra que el sistema no es complejo, quedando demostrada la calidad del diseño. Los resultados obtenidos son positivos según esta métrica, como se puede ver en la siguiente tabla.

Umbral	Tamaño	Cantidad de Clases
<= 20	Pequeño	10
>20 y <= 30	Medio	5
>30	Grande	0

Tabla 2. Cantidad de clases por tamaño.

Así se concluye que la Capa del Negocio cuenta con 15 clases, para un promedio de cantidad de atributos de 0.73 y un promedio de cantidad de operaciones de 15.47. Para más información sobre la descripción de estas clases (Ver Anexo13).

3.2 .2 Árbol de Profundidad de Herencia (APH).

Esta métrica se define como la longitud máxima desde el nodo hasta la raíz del árbol. A medida que crece el APH, es más probable que las clases de niveles inferiores hereden muchos métodos. Esto da lugar a posibles dificultades cuando se intenta predecir el comportamiento de una clase.
Una jerarquía de clases profunda con un valor grande de APH, lleva también a una mayor complejidad del diseño. Por el lado positivo, los valores grandes de APH implican que se pueden reutilizar muchos métodos.
Por su parte, algunos autores como Lorenz y Kidd sugieren que un umbral de 6 niveles como indicador es un abuso en la herencia en distintos lenguajes de programación.

Resultado.
A partir de los datos obtenidos al aplicar la métrica APH se obtuvo que el nivel más alto de herencia entre las clases del diseño es de: 1 nivel de profundidad, por lo que queda demostrado que el diseño no es complejo, ya que existe un bajo acoplamiento entre las clases y no es difícil su mantenimiento, pues el nivel de profundidad de la herencia es el mínimo que puede existir en la relación entre clases, este resultado conlleva a que el diseño realizado tiene calidad.

3.2.3 Número de Operaciones Redefinidas para una Sub-Clase (NOR).

Según Pressman gran conocedor del tema, los valores grandes para el NOR, generalmente indica un problema en el diseño, o sea si el NOR es grande el diseñador ha violado la abstracción representada por la superclase. Esto provoca una débil jerarquía de clases y un software orientado a objetos, que puede ser difícil de probar y modificar. [8]
Esta es una de las métricas aplicadas en el proyecto Sistema de Gestión Fiscal para medir la calidad del diseño propuesto.
Resultado.
A partir de los datos obtenidos después de aplicarle al sistema la métrica NOR se obtuvieron que de 15 subclases con las que cuenta el sistema en la capa del negocio ninguna de las 15 subclases reemplazan operaciones de las superclases de la que heredan, estas solo reutilizan los métodos de la clase padre pero no se redefinen ninguno de los métodos, en caso que falte hacer alguna operación nueva, se crea un nuevo método, por lo que el diseño propuesto posee gran calidad pues así se demostró al aplicar la métrica en el diseño realizado.

3.3 Resultados de las Pruebas de Caja Negra.

Las pruebas de caja negra se centran en lo que se espera de un módulo, intentan encontrar casos en que el módulo no cumple con su especificación, estas pruebas se apoyan en la especificación de requisitos del módulo, sin preocuparse de lo que pueda estar haciendo el módulo por dentro. Las pruebas de caja negra son sinónimo de:

- Pruebas de caja opaca.

- Pruebas funcionales.
- Pruebas de entrada/salida.
- Pruebas inducidas por los datos.

Según lo definido por Pressman, las pruebas de caja negra se llevan a cabo sobre la interfaz del software. Se trata de demostrar que las funciones del software son operativas, que las entradas se manejan de forma adecuada y que se produce el resultado esperado.
Las pruebas de caja negra buscan encontrar errores en cinco categorías [8]

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a base de datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

Estas pruebas de caja negra han ayudado al grupo de desarrolladores del proyecto Sistema de Gestión Fiscal a convencerse de que el programa hace lo que quiere y desea el cliente, cumpliendo de esta forma con los requisitos que debe realizar la aplicación.
A continuación las pruebas realizadas al módulo de Gestión de Cuadro del Subsistema de Gestión de Cuadro y Personal de Apoyo del proyecto Sistema de Gestión fiscal.

Módulo Gestión de Cuadro
Descripción General
Este caso de uso consiste en gestionar la creación de los expedientes de los cuadros a partir de la introducción de los **datos personales del cuadro**. Luego de introducir los datos del cuadro se guarda la información.
Condiciones de Ejecución:
Que el usuario tenga permisos para realizar la creación del cuadro.
Que el cuadro no exista en la Base de Datos.

Nombre de Sección	Escenarios de Sección	Descripción de funcionalidad	Flujo Central
SC 1: Registrar un Cuadro.	EC1:Registrar Datos Personales del Cuadro	El sistema muestra los campos que deben ser llenados, después de ser llenados guarda los datos.	El usuario accede a la pestaña de Datos Personales, el sistema muestra los campos que se deben llenar para registrar un cuadro: Número de Referencia Nombre, Primer Apellido. Segundo Apellido. Carnet de Identidad. Edad.

			Sexo. Etnia (blanco, negro). Militancia (PCC, UJC, No Militante). Provincia donde trabaja. Fiscalía donde trabaja (Todas las provincias con sus municipios y el Órgano Provincial respectivo a cada provincia, se incluye el Municipio especial Isla de la Juventud, en caso de Ciudad Habana se incluye el Órgano Central). - Dirección particular. - Provincia donde reside. - Municipio donde reside. -Teléfono. - Nombre de los padres. El usuario llena todos los campos y presiona el botón Examinar para buscar la foto del cuadro en la computadora, El sistema muestra la ventana donde el usuario seleccionará el lugar donde se encuentra la foto del cuadro. Busca en la ventana que aparece la foto deseada y presiona el botón Guardar para guardar toda la información almacenada. El sistema chequea que los campos Nombre, Primer Apellido, Segundo Apellido, Carnet de Identidad, Provincia donde trabaja, Municipio donde trabaja, Foto del Cuadro y Correo Electrónico, estén llenos. En caso de que todos estos campos estén llenos y guarda la información. 3.2 Calcula la edad del cuadro. 3.3 Crea un nuevo usuario para

			el cuadro registrado. 3.4 Crea una contraseña para el cuadro. 3.5 Envía un correo de notificación al cuadro con su usuario y contraseña. 3.6 Guarda la imagen del cuadro registrado. 3.7 Guarda las acciones realizadas por el usuario que está registrando los datos en el sistema (con la siguiente estructura): nombre del usuario, la fecha en que realizó la acción y mensaje (este caso sería): "Registró datos personales del cuadro: [nombre del cuadro]". 3.8 El caso de uso termina.
	EC 2: Error en la entrada de datos	El sistema verifica que la entrada de datos sea correcta.	El usuario no llena los campos obligatorios y/o no entra exactamente 11digitos en el carnet de identidad. El sistema verifica que hay campos obligatorios sin llenar por tanto lanza un mensaje de error. 3.2 El sistema verifica que el número del carnet de identidad no tiene exactamente los 11 dígitos y muestra un mensaje de error.
	EC3: Cancelar	Detener la acción.	Presionar el botón cancelar.
	EC4: Cerrar	Cierra la aplicación.	1: Salir 2: Presionar el botón cerrar.

SC 1: Registrar Cuadro

Id del escenario	Escenario	Nombre y apellidos	Número de carnet identidad	Edad	Dirección	Fecha	Correo	Respuesta del Sistema	Resultado de la Prueba

EC 1	Registrar Datos Personales del Cuadro	V	V	V	V	V	El sistema da la opción de registrar los datos personales del cuadro.	La prueba resultó satisfactoria.
EC 2	Error en la entrada de datos.	V	I	I	V	I	El sistema emite un mensaje para que llene los campos obligatorios.	La prueba resultó satisfactoria.
E C3	Cancelar	NA	NA	NA	NA	NA	El sistema cancela la operación.	La prueba resultó satisfactoria.
EC 4	Cerrar	NA	NA	NA	NA	NA	El sistema cierra la aplicación.	La prueba resultó satisfactoria.

[Las celdas de la tabla contienen V, I, o N/A. V indica válido, I indica inválido, y N/A que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante.]

Descripción General

Este caso de uso consiste en modificar los **datos laborales de los cuadros**, los cuales son trayectoria laboral, los viajes que ha realizado y los motivos de los mismos, las sanciones que se le han aplicado al fiscal, el dominio de idiomas y por último las condecoraciones para la creación del expediente de los mismos.

Condiciones de Ejecución:

Que el usuario tenga acceso a esta parte del sistema.

Nombre de Sección	Escenarios de Sección	Descripción de funcionalidad	Flujo Central

SC 1: Registrar Datos Laborales del Cuadro	EC 1: Registrar datos laborales del cuadro	El sistema muestra los datos del cuadro y los que se deben llenar los verifica y guarda la información.	El sistema muestra la experiencia del fiscal resultado de la resta entre la Fecha actual y la Fecha de inscripción, introducidas con anterioridad en la interface datos personales. 1.3- El sistema muestra todos los campos que se deben llenar para registrar los datos laborales del cuadro: -País. - Motivo - Fecha en la cual viajó -Esfera de trabajo (una o varias). - Sanciones Políticas (Amonestación, Suspensión Temporal de Derecho, Separación de Cargo, Separación, Expulsión). - Sanciones Administrativas (Amonestación privada, Amonestación Pública, Democión, Multa). - Sanciones Judiciales (Multas, Privación de Libertad, Subsidiarias). - Fecha de la sanción. - Idioma. - Habla (B, R, M). - Lee (B, R, M). - Escribe (B, R, M). - Nivel alcanzado (1er nivel, 2do nivel, 3er. nivel, 4to nivel, 5to nivel). - Año de graduado. El Responsable de Cuadro introduce los datos

			correspondientes a los Viajes al Exterior realizados: País, Motivo por el cual viajó y Fecha en la cual viajó y da clic en el botón Agregar. 2.1- El sistema visualiza en la tabla Viajes al exterior, los datos que fueron adicionados. En caso de haber realizado más viajes, ir al paso 2. El Responsable de Cuadro introduce los datos correspondientes al Idioma, Habla, Lee, Escribe, Nivel alcanzado y Año de graduado. El sistema visualiza en la tabla Idioma, los datos que fueron adicionados. En caso de hablar más idiomas, ir al paso 3. El Responsable de Cuadro introduce todos los restantes datos: Esfera de trabajo (una o varias), Sanciones Políticas, Sanciones Administrativas, Sanciones Judiciales y Fecha de la sanción. El Responsable de Cuadro selecciona la opción Ver más. Muestra todas las condecoraciones. Que pueden ser: -Distinción ¨ 28 de Septiembre¨(CDR) -Distinción ¨ Enrique Hart (25 años Sector Público). -Medalla ¨Hazaña Laboral¨.

			-Medalla ¨40 Aniversario de las FAR¨ Medalla ¨Alfabetización¨. -Medalla ¨ Producción y Defensa¨ Distinción ¨Antelo Regalado¨ Distinción ¨ 23 de Agosto¨(FMC) -Sello por Años de Servicio en la Fiscalía -Sello Fundador de la Fiscalía -Medalla ¨ XX Aniversario¨ -Medalla ¨ XXX Aniversario ¨ (FAR) -Medalla ¨ 50 Aniversario de las FAR¨ -Medalla ¨ Combatiente Internacionalista¨ (FAR) -Medalla ¨ Trabajador Internacionalista¨. -Medalla ¨ Combatiente de Lucha. Clandestina¨ (FAR). -Medalla ¨ Combatiente del Ejército Rebelde¨ (FAR). -Medalla ¨ Por la Seguridad del Orden Interior¨ (FAR). -Medalla ¨ Servicio Distinguido¨ (FAR) Medalla ¨ Victoria de Girón¨ (FAR). -Medalla ¨ Por la Defensa de la Patria y la Unidad del Barrio¨ (CDR). -Vanguardia Nacional. -Vanguardia Provincial. -Vanguardia Municipal -Vanguardia bueno. El Responsable de Cuadro selecciona las condecoraciones del fiscal y pone la fecha en la cual recibió cada una de ellas. El

			Responsable de Cuadro selecciona la opción de guardar los datos introducidos. El sistema guarda la información. Guarda las acciones realizadas por el usuario en el sistema, con la siguiente estructura: [nombre del usuario], [fecha en que realizó la acción] y [mensaje], que en este caso sería "Registró datos laborales del cuadro: [nombre del cuadro]". Se termina el caso de uso.
	EC 2: Espacios sin llenar	Muestra un mensaje de error donde expresa que existen campos sin llenar.	Muestra un mensaje de error donde expresa que existen campos sin llenar. Muestra un mensaje de error donde expresa que existen campos sin llenar. Muestra un mensaje de error donde expresa que existen campos sin llenar.
	EC3: Cancelar	Detener la acción.	Presionar el botón cancelar.
	EC4: Cerrar	Cierra la aplicación.	1: Salir 2: Presionar el botón cerrar.

SC 1: Registrar Datos Laborales del Cuadro

Id del escenario	Escenario	Viajes	Sanciones	Idiomas	Esfera de Trabajo	Condecoraciones	Respuesta del Sistema	Resultado de la Prueba

EC 1	Registrar datos laborales del cuadro	V	V	V	V	V	El sistema da la opción de registrar los datos laborales del cuadro.	La prueba resultó satisfactoria.
EC 2	Espacios sin llenar.	I	I	I	NA	NA	El sistema emite un mensaje para que llene los campos obligatorios.	La prueba resultó satisfactoria.
E C3	Cancelar	V	V	V	V	V	El sistema cancela la operación.	La prueba resultó satisfactoria.
EC 4	cerrar	NA	NA	NA	NA	NA	El sistema cierra la aplicación.	La prueba resultó satisfactoria.

[Las celdas de la tabla contienen V, I, o N/A. V indica válido, I indica inválido, y N/A que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante.]

Descripción General

Este caso de uso consiste en introducir los datos de la evaluación del cuadro, los cuales están relacionados con la reserva del cuadro, si es miembro de la misma o no, en caso de ser miembro se introduce el cargo para el cual se prepara y para cuando se piensa promover el mismo, además se introduce la evaluación del cuadro en un trimestre del año y si no es evaluado los motivos, se concluye el caso de uso guardando toda la información registrada.

Condiciones de Ejecución:

Que el usuario tenga permisos para acceder a esta parte del sistema.

Nombre de Sección	Escenarios de Sección	Descripción de funcionalidad	Flujo Central
SC1: Registrar datos de la evaluación del cuadro.	EC1: Registrar datos de la evaluación del cuadro.	El Responsable de Cuadro registra los datos de la evaluación del cuadro.	1. Acceder a la pestaña de Datos de Reserva y Evaluación. 2. Se muestra en una tabla los datos de las evaluaciones de los últimos tres años y muestra los campos que debe llenar el usuario: -Integra la reserva (Si, No). -Cargo para el que se prepara que pueden ser:

			Fiscal General Vice fiscal General Fiscal Jefe Dirección Fiscal Jefe Departamento Fiscal Jefe Provincial Vice fiscal Jefe Provincial Fiscal Jefe Municipio Vice fiscal Jefe Municipio Fiscal provincial Fiscal municipal -Promover (A largo plazo, A mediano plazo y a corto plazo). -Trimestre (Primer Trimestre, Segundo Trimestre, Tercer Trimestre). - Evaluación (MB, B, R, M). - Periodo de evaluación. - Motivo por el cual no fue evaluado (bajas, licencia de maternidad, certificado médico prolongado, fuera de provincia, injustificada). 3. Seleccionar que el Cuadro integra la reserva. 4. Se habilitan los campos: -Cargo para el que se prepara. -Promover(A largo plazo, A mediano plazo y a corto plazo). 5. Introducir los datos correspondientes a los campos habilitados y el resto de los campos, los cuales son: -Trimestre (Primer Trimestre, Segundo Trimestre, Tercer Trimestre). - Evaluación. - Periodo de evaluación. - Motivo por el cual no fue evaluado (bajas, licencia de maternidad, certificado médico prolongado, fuera de provincia, injustificada). - Observaciones. Y seleccionar la opción de guardar. 6. Se verifica que los datos obligatorios se

			hayan llenado, los cuales son: si integra la reserva, cargo para el que se prepara y promover. 7. Se guardan las acciones realizadas.
	EC 2: Flujo alterno	Consiste en detectar el error.	1: Seleccionar que el Cuadro no integra la reserva. 2: Se deshabilitan los campos: -Cargo para el que se prepara. 3: Se verifica que los datos obligatorios no fueron introducidos y muestra un mensaje de error. 4: Llenar los datos obligatorios que le faltaron por llenar y seleccionar la opción Guardar. 5: Se guarda la información. 4.2 Se guardan las acciones realizadas
	EC3: Cancelar	Detener la acción.	Presionar el botón cancelar.
	EC4: Cerrar	Cierra la aplicación.	1: Salir 2: Presionar el botón cerrar.

SC1: Registrar datos de la evaluación del cuadro.

Id del escenario	Escenario	Reserva	Trimestre	Evaluaciones	Periodo de evaluación	Observaciones	Respuesta del Sistema	Resultado de la Prueba
EC 1	Registrar datos de la evaluación del cuadro	V	V	V	NA	NA	El sistema da la opción de registrar los datos de la reserva y la evaluación de los cuadro.	La prueba resultó satisfactoria.
EC 2	Flujo alterno	V	I	I	NA	NA	El sistema emite un mensaje para que llene los campos obligatorios.	Se escribe el resultado que se obtiene al realizar la prueba.

EC3	Cancelar	V	V	V	V	V	El sistema cancela la operación.	La prueba resultó satisfactoria.
EC 4	Cerrar	NA	NA	NA	NA	NA	El sistema cierra la aplicación.	La prueba resultó satisfactoria.

[Las celdas de la tabla contienen V, I, o N/A. V indica válido, I indica inválido, y N/A que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante.]

Descripción General

Este caso de uso permite al Responsable de Cuadro del sistema registrar los permisos de los usuarios. El Responsable de Cuadro asigna los niveles de permisos al sistema.

Condiciones de Ejecución:

El usuario debe tener permiso de para acceder a esta parte del sistema.

Nombre de Sección	***Escenarios de Sección***	***Descripción de funcionalidad***	***Flujo Central***
SC1: Registrar Permisos.	EC 1: El Responsable de Cuadro accede a la interfaz del Módulo Gestión de Cuadro y Personal de Apoyo para registrar los permisos de un cuadro.	El sistema muestra la interfaz de asignar permisos.	El caso de uso comienza cuando el Responsable de Cuadro accede a la interfaz del Módulo Gestión de Cuadro y Personal de Apoyo para registrar un cuadro. Sistema muestra el Módulo completo de Gestión de Cuadro y Personal de Apoyo. El Responsable de Cuadro accede a la interfaz de asignar permisos. El sistema muestra una ventana con los siguientes datos -nombre -cargo -permisos El Responsable de Cuadro puede marcar los permisos que desee

			asignarle al usuario y desmarcar los permisos que desee quitarle. El Responsable de Cuadro presiona el botón guardar. El sistema guarda los cambios. El sistema guarda las acciones realizadas por el usuario en el sistema, con la siguiente estructura: [nombre del usuario], [la fecha en que realizó la acción] y [mensaje], que en este caso sería "Registró permisos del cuadro: [nombre del cuadro]". Se termina el caso de uso.
	EC 2: El Responsable de Cuadro presiona el botón Finalizar sin guardar los cambios.	El sistema finaliza la operación sin guardar los datos.	El Responsable de Cuadro presiona el botón finalizar sin guardar los cambios.

SC1: Registrar Permisos.

Id del escenario	Escenario	Permisos Asignados	Respuesta del Sistema	Resultado de la Prueba
EC 1	Registrar Permisos del Cuadro.	NA	Se escribe el resultado que se espera al realizar la prueba.	Se escribe el resultado que se obtiene al realizar la prueba.

EC 2	Finalizar	NA	El sistema finaliza la operación.	La prueba resultó satisfactoria.

[Las celdas de la tabla contienen V, I, o N/A. V indica válido, I indica inválido, y N/A que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante.]

3.4 Conclusiones.

En este capítulo se trataron los temas relacionados con las métricas aplicadas al sistema logrando de esta forma obtener un producto robusto y flexible que garantiza mayor calidad en los proceso fiscales proporcionando mayor comodidad entre sus funcionarios a la hora de realizar el trabajo, también se realizaron las pruebas de Caja Negra que demostraron que el sistema cumple con las exigencias que se piden por parte del cliente.

CONCLUSIONES

Con la realización del presente trabajo se arriba a las siguientes conclusiones:

- Los patrones de arquitectura, de diseño y las métricas aplicadas, dieron lugar a un sistema robusto y flexible, aspectos muy importantes en el objetivo de este trabajo.
- Como resultado del diseño del sistema se obtuvieron los Diagramas de Interacción y los Diagramas de Clases imprescindibles para el buen desarrollo de una solución informática con calidad.
- Con los resultados obtenidos del diseño del sistema se determinó el Modelo de Componentes y el Modelo de Despliegue para el sistema, artefactos generados en el flujo de implementación.
- Las herramientas utilizadas para realizar el Diseño y la Implementación del sistema fueron mayormente libres, cumpliendo de esta forma con la política que sigue la Fiscalía General de la República de Cuba de lograr la soberanía tecnológica.
- Con la realización de las Pruebas de Caja Negra al sistema, quedó demostrado que el software cumple con todas las funcionalidades que pide el cliente.
- Como resultado general se obtuvo el Diseño y la Implementación del Subsistema Gestión de Cuadro y Personal de Apoyo del Proyecto Sistema de Gestión Fiscal, dándole cumplimiento de esta forma al objetivo general de este trabajo.

RECOMENDACIONES

- ACTUALIZAR LA VERISION DEL SYMFONY A VERSIONES SUPERIORES CADA 1 AÑO, PUES ESTE MEJORA CONSTANTEMENTE LA COMPATIBILIDAD CON LOS DISTINTAS VERSIONES DE PHP, MEJORAS DE LA SEGURIDAD, E INTEGRA UNA SERIE DE PLUGINS LOS CUALES AUMENTAN LA COMODIDAD Y LA FUNCIONALIDAD DEL FRAMEWORK.

BIBLIOGRAFÍA

[1] **León Hernández, R. A. y González Coello, S. (2002).** *El Paradigma Cuantitativo de la Investigación Científica.* Ciudad de La Habana: Editorial Universitaria, 2002. 959-16-0343-6.

[2] Home. Estudios y Proyectos Informáticos. [En Línea] ABOGest. [Citado el: 2 de Octubre de 2008.] http://www.abogest.com/index.htm

[3] Home**.** Brindys Software. [En Línea] GEDEX. [Citado el: 2 de Octubre de 2008.] http://www.brindys.com/gedex/

[4] Home**.** Venezuela Jurídica [En Línea] Compilación Jurídica. [Citado el: 2 de Octubre de 2008.] http://www.venezuelajuridicaenlinea.com/

[5] **Schmuller, Joseph. 2000.** Aprendiendo UML en 24 horas. [Pdf] Mexico: PEARSON EDUCACION, 2000.

[6] **Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000.** El Proceso Unificado de Desarrollo de Software. Madrid: Addison Wesley, 2000.

[7] **Craig Larman. 1999.** *UML y Patrones. Introducción al Análisis y Diseño orientado a objetos.* México: Prentice Hall, 1999. 970-17-0261-1.

[8] **Roger S. Pressman. 1998.** *Ingeniería de Software. Un Enfoque Práctico.* 1998.

[9] **Erich Gamma, y otros. 1994.** *Design Patterns: Elements of Reusable Object-Oriented Software.* China: KevinZhang, 1994.

[10] Home. *Visual Paradigm.* [En línea] Visual Paradigm. [Citado el: 10 de Noviembre de 2008.] http://www.visual-paradigm.com.

[11] Home. *Grupo Soluciones GSInnova.* [En línea] Rational. [Citado el: 1 de Noviembre de 2008.] http://www.rational.com.ar/.

[12] Home**.** Métricas Técnicas del Software*.* [En línea] Métricas. [Citado el: 10 de Diciembre de 2008.] http://www.uv.mx/asumano/Métricas%20Técnicas.pdf

BIBLIOGRAFÍA

[13] **Grady Booch. 1998.** *OBJECT-ORIENTED ANALYSIS AND DESIGN.* California: ADDISON-WESLEY, 1998. 0-8053-5340-2.

[14] Home. Zend Framework. [En línea] Zend Framework. [Citado el: 10 de Noviembre de 2008.] http://framework.zend.com/

[15] Home. Spring .NET. [En línea] Spring .NET. [Citado el: 10 de Noviembre de 2008.] http://www.springframework.net/
http://tuxpuc.pucp.edu.pe/content/view/745/12/

[16] Home.symfony.es. [En línea] symfony. [Citado el: 10 de Noviembre de 2008.] http://www.symfony.es/

[17] Home. symphony. [En línea] symphony. [Citado el: 10 de Noviembre de 2008.] http://www.symfony-project.org/

[18] **Carlos Reynoso y Nicolás Kiccillof. 2004.** *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Buenos Aires: UNIVERSIDAD DE BUENOS AIRES, 2004, Vol. Versión 1.0.

[19] **J. Bosch. 2000**. Designand Use of Software Architectures. California: Addison Wesley.

[20] Home**.** Introducción a PostgreSQL. [En línea] PostgreSQL. [Citado el: 10 de Diciembre de 2008.]
http://www.postgresql.org.pe/articles/introduccion_a_postgresql.pdf

[21] **Lorenz y Kidd. 1994.** *Métricas de Lorenz y Kidd.* 1994.

[22] **Chidamber y Kemne. 1994.** *Métricas CK.* 1994.

[23] Home. Métricas para Sistemas Orientados a Objetos. [En línea] Métricas. [Citado el: 10 de Diciembre de 2008.]
http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/gonzalez_d_h/capítulo6.pdf

[24] Home. eclipsecon™2009. [En línea] Eclipse. [Citado el: 10 de Diciembre de 2008.] http://www.eclipse.org/

[25] **Tom Archer. 2001.** *A Fondo C#.* Madrid: McGraw-Hill/Interamericana de España, 2001. 0-7356-1288-9.

[26] **Ojeda, Francisco Charte. 2002.** *Visual C#. NET.* Madrid: Ediciones Anaya Multimedia, 2002. 84-415- 1392-9.

BIBLIOGRAFÍA

[27] Home. Visual Studio. [En línea] Visual Studio. [Citado el: 10 de Enero de 2009.] http://msdn.microsoft.com/es-es/vstudio/default.aspx

[28] **Robles, Gregorio; Grupo de Sistemas y Comunicaciones. 2003.** Programación eXtrema (y software libre). Madrid: s.n.

[29] **Molpeceres, Alberto. 2002.** Procesos de desarrollo: RUP, XP, FDD.

[30] Home. Metodología XP Vs. Metodología RUP [En línea] XP vs RUP. [Citado el: 10 de Octubre de 2008.] http://metodologíaxpvsmetodologíarup.blogspot.com/

[31] Home**.** PHP en castellano. [En línea] ¿Por qué elegir PHP? [Citado el: 14 de noviembre de 2008.] http://www.programación.net/php/articulo/porquephp/.

[32] Home. Guía de Iniciación al Lenguaje JAVA. [En Línea] Java [Citado el: 14 de noviembre de 2008.] http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/

[33] **Bjarne Stroustrup. 2000**. *El lenguaje de programación C++*. Madrid: Addison-Wesley Pub Co; Tercera edición; ISBN 0-201-70073-5

[34] Home. Microsoft SQL Server. [En Línea] SQL Server [Citado el: 10 de Diciembre de 2008.] http://www.microsoft.com/spain/sql/default.mspx

[35] **Kruchten, Philippe. 1995**. "Architectural Blueprints--The 4+1 View Model of Software Architecture". IEEE Software, Institute of Electrical and Electronics Engineers. pp. 42-50

[36] **Shaw, Mary y Clements, Paul. 1996.** *A field guide to Boxology: Preliminary classification of architectural styles for software systems.* Pennsylvania, Estados Unidos de América: Computer Science Department and Software Engineering Institute, Carnegie Mellon University.

[37] **Shaw, Mary y Garlan, David. 1996.** *Software Architecture: Perspectives on an emerging discipline.* Upper Saddle River: Prentice Hall.

ANEXOS

Anexo 1. Diagramas de Secuencia del Módulo de Gestión de Cuadro.

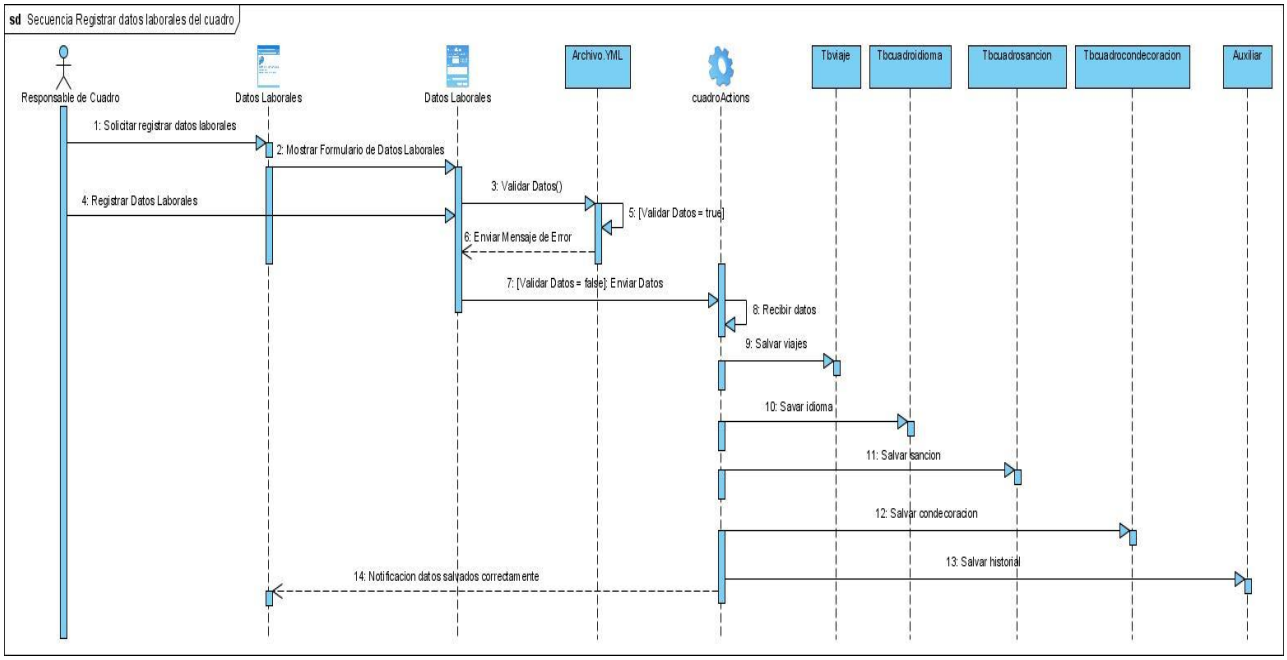

Figura 1. Secuencia: Registrar Datos Laborales del Cuadro.

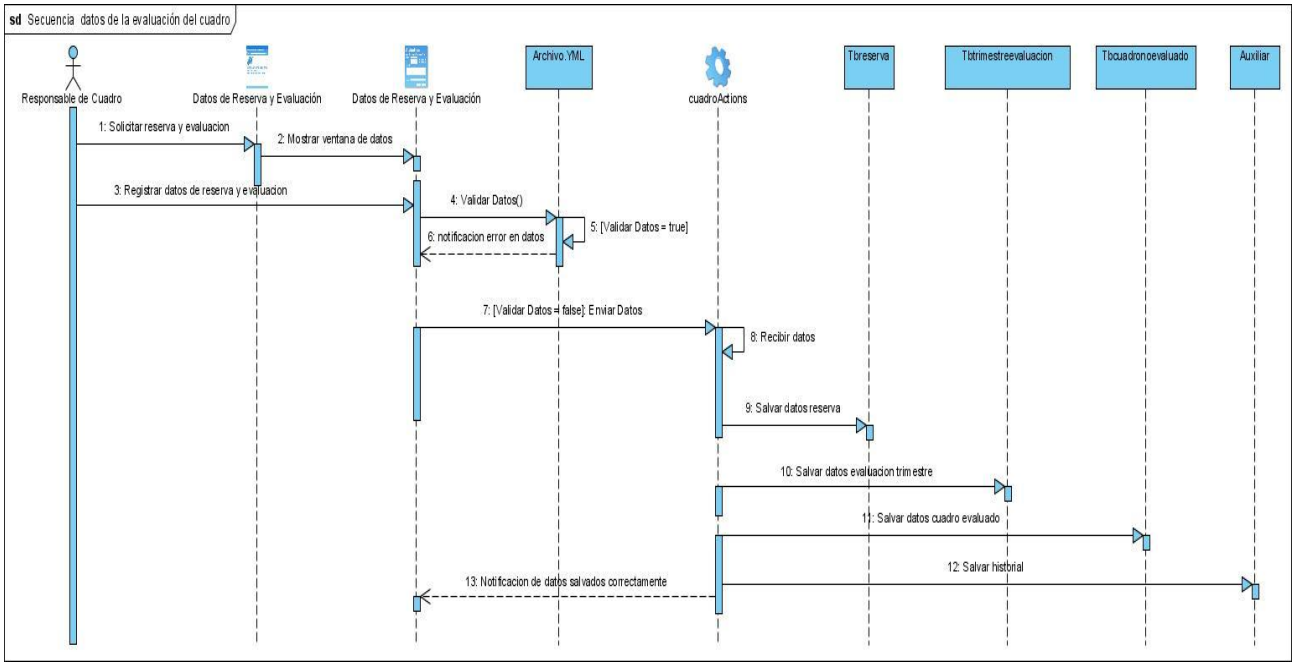

Figura 2. Secuencia: Registrar Datos de la Evaluación del Cuadro.

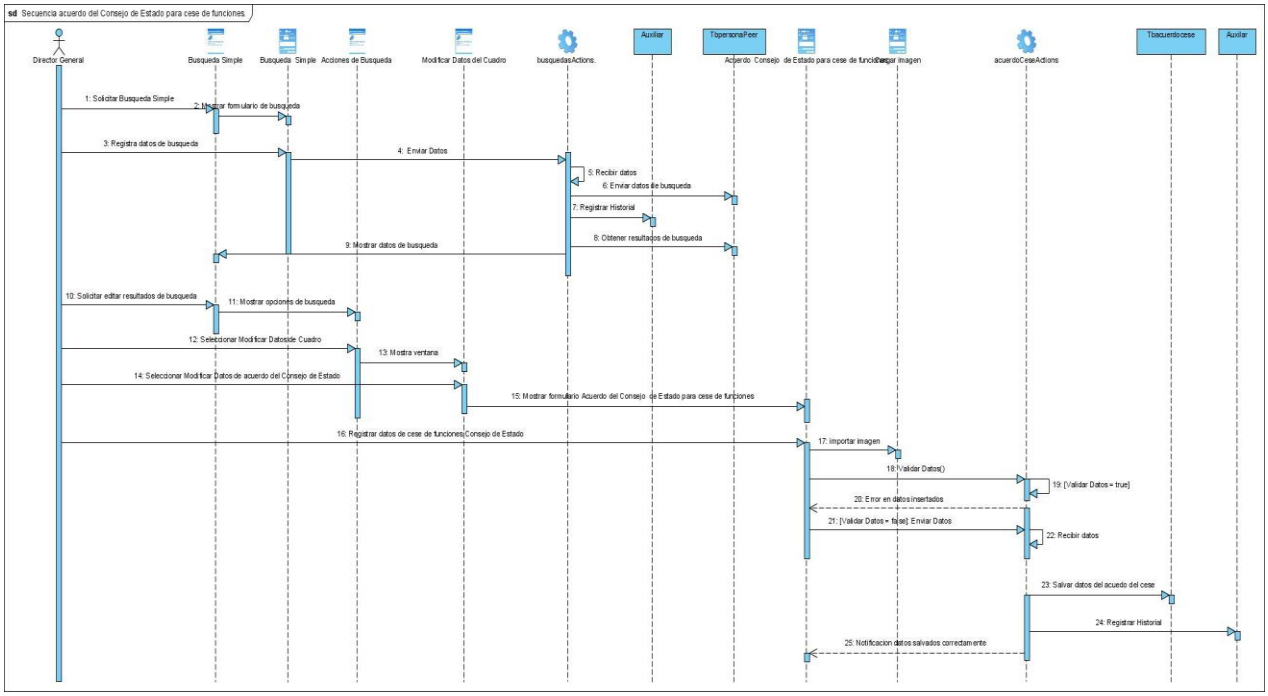

Figura 3. Secuencia: Registrar Acuerdo del Consejo de Estado para Cese de Funciones.

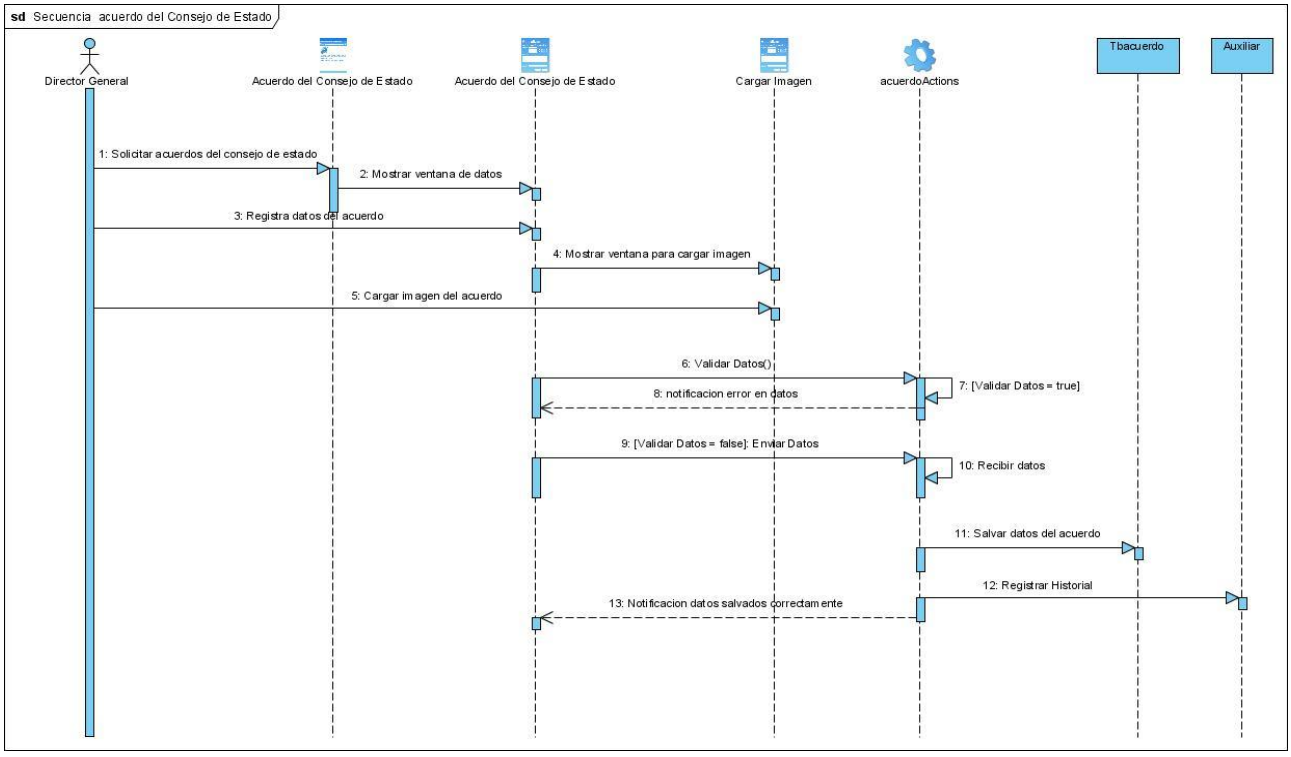

Figura 4. Secuencia: Registrar Acuerdo del Consejo de Estado.

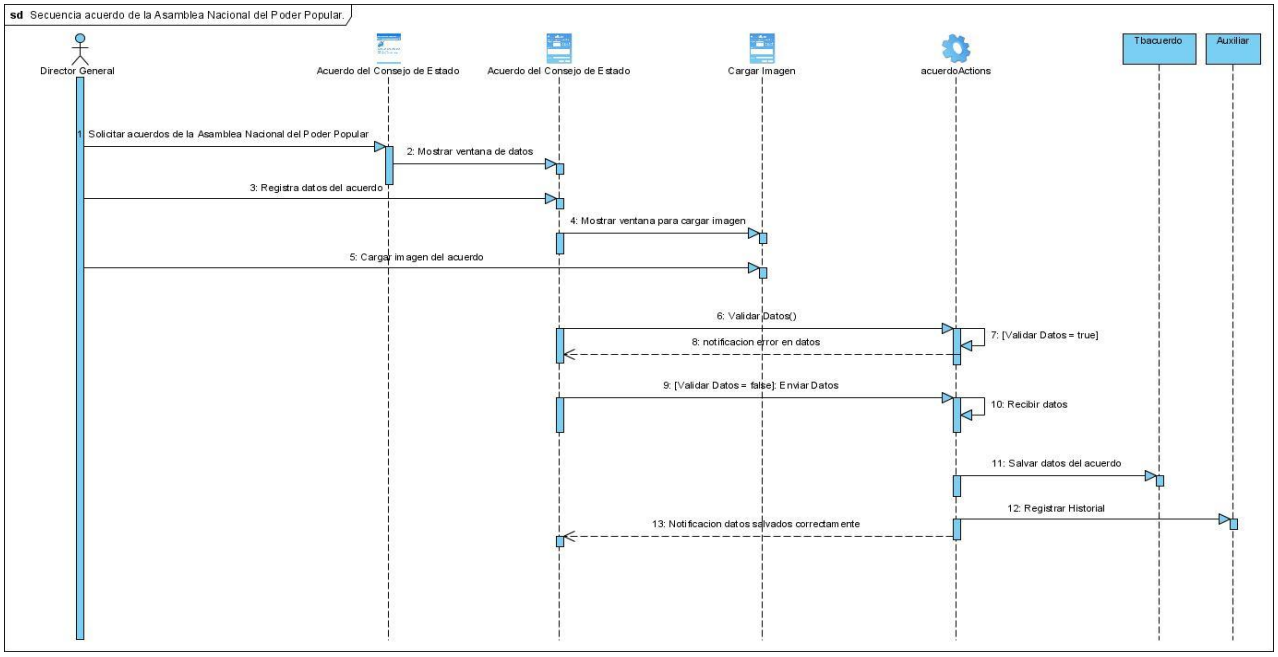

Figura 5. Secuencia: Registrar Acuerdo de la Asamblea Nacional del Poder Popular.

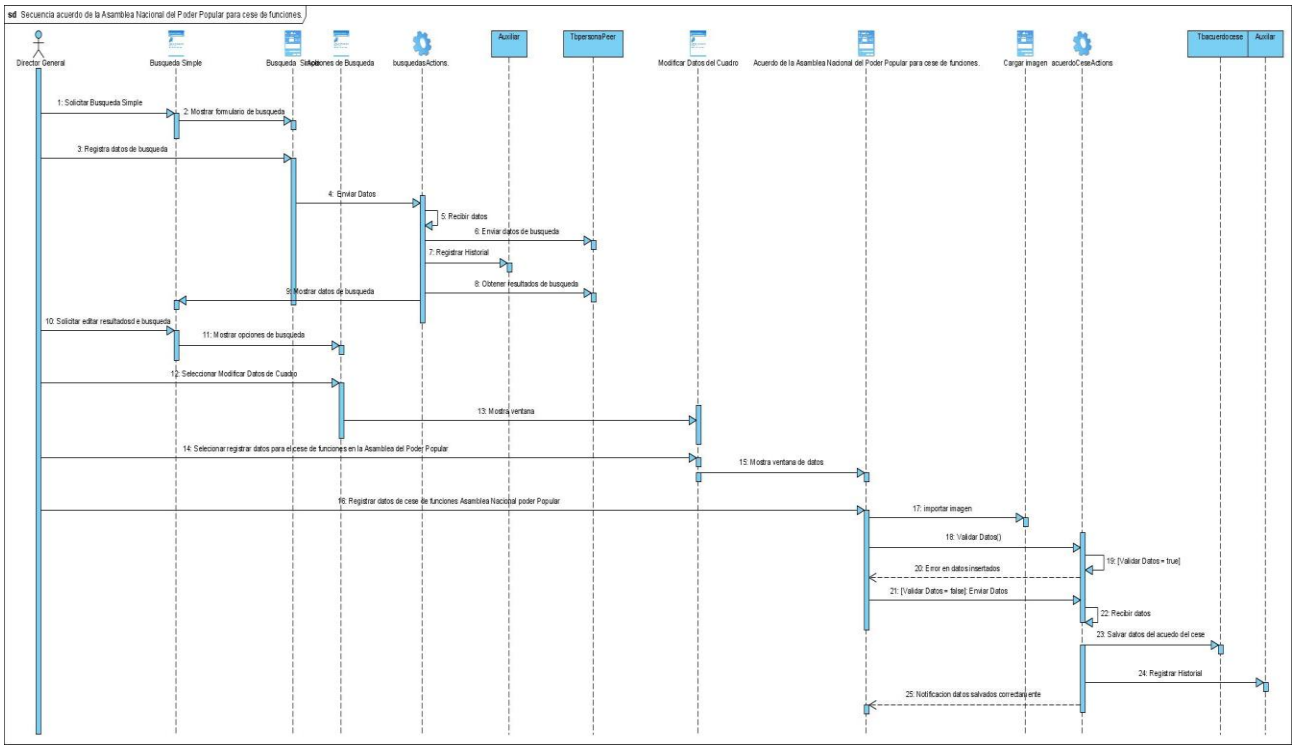

Figura 6. Secuencia: Registrar Acuerdo de la Asamblea Nacional del Poder Popular para Cese de Funciones.

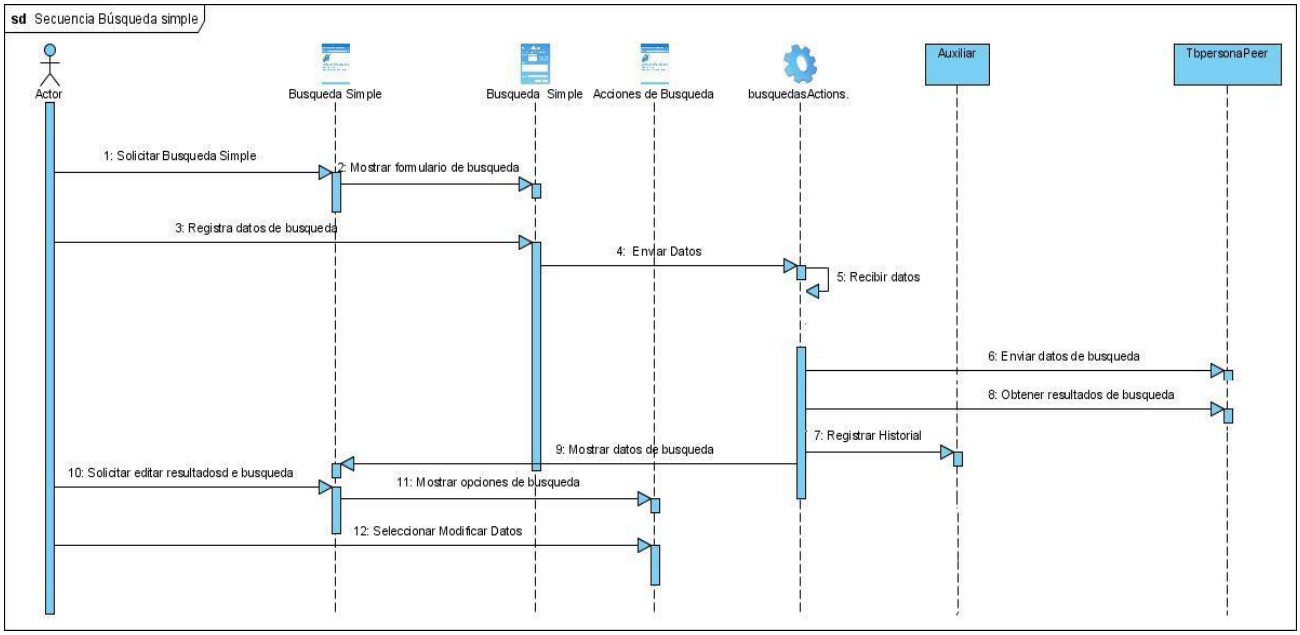

Figura 7. Secuencia: Búsqueda Simple.

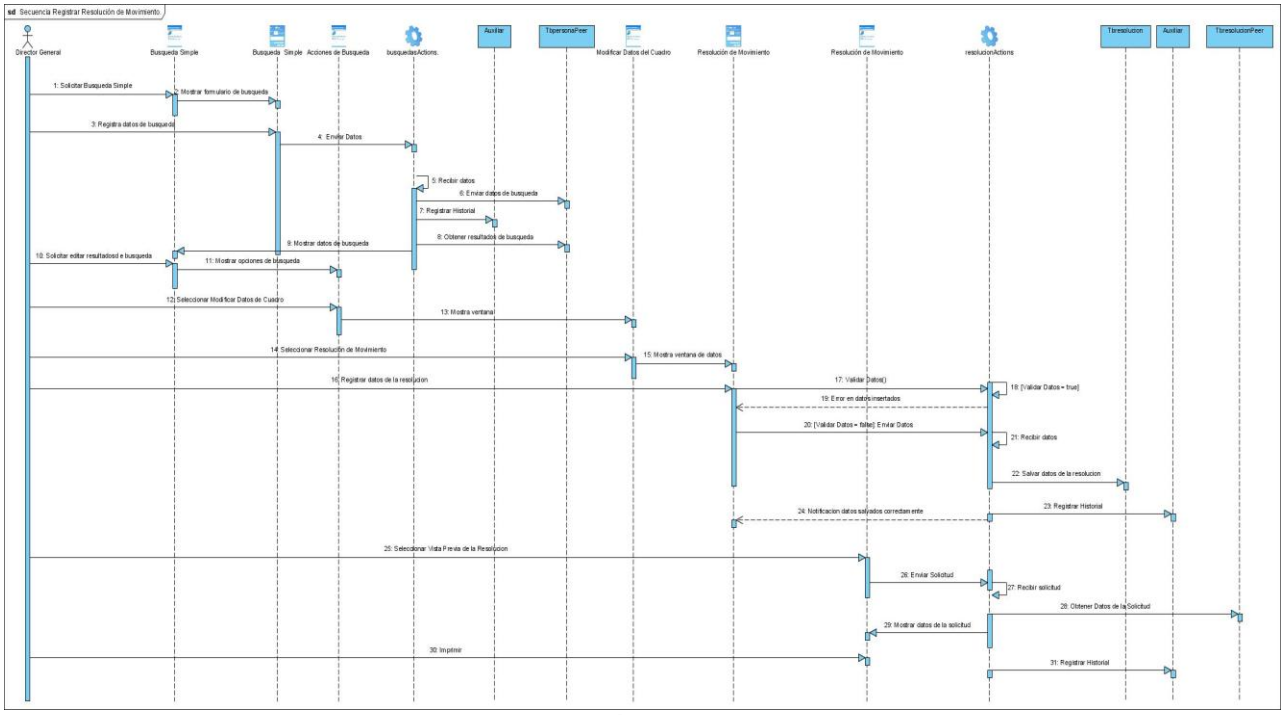

Figura 8. Secuencia: Registrar Resolución de Movimiento.

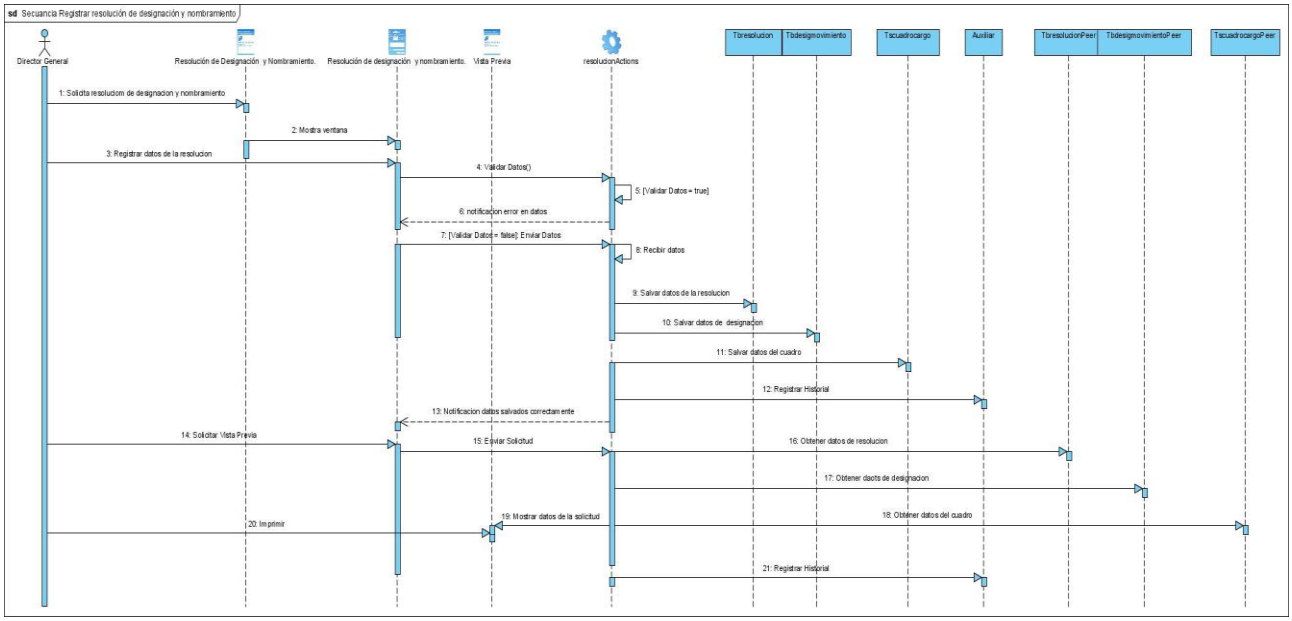

Figura 9. Secuencia: Registrar Resolución de Designación y Nombramiento.

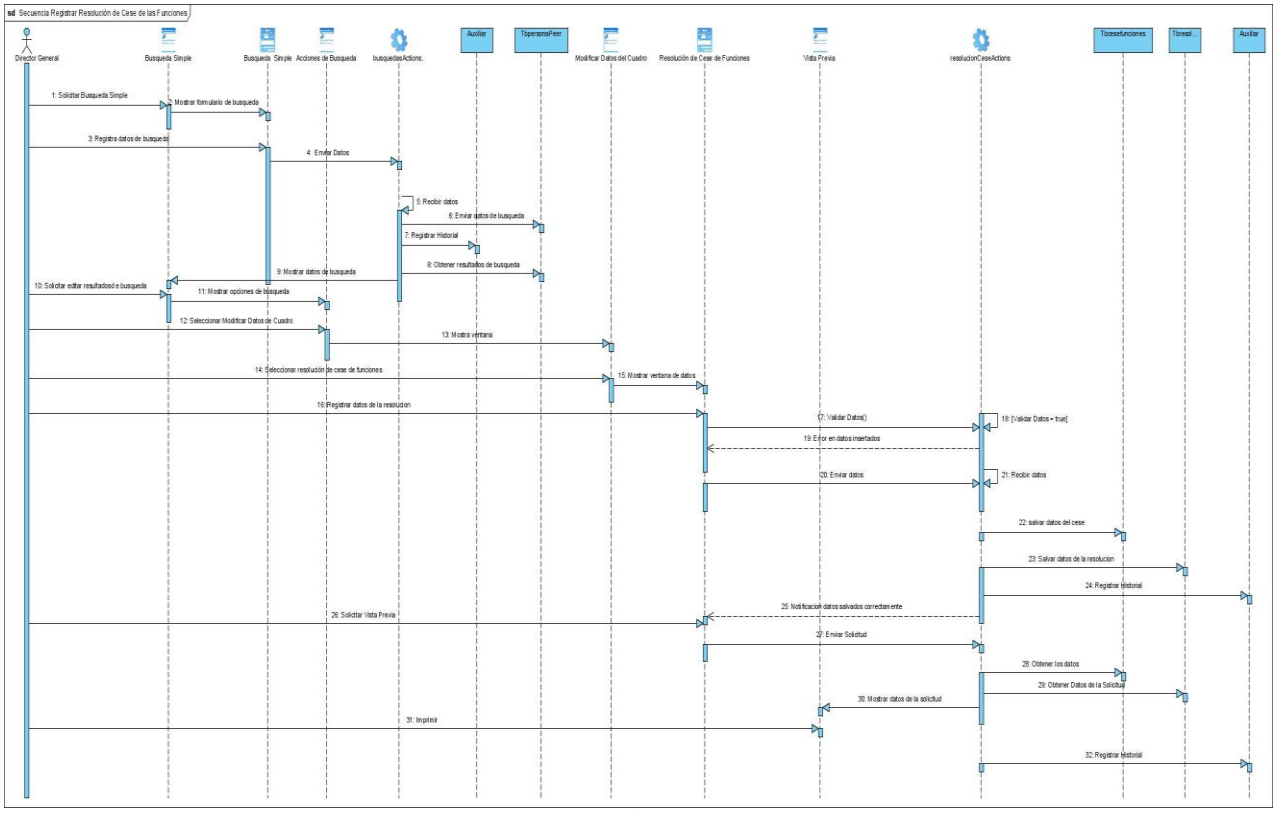

Figura 10. Secuencia: Registrar Resolución de Cese de las Funciones.

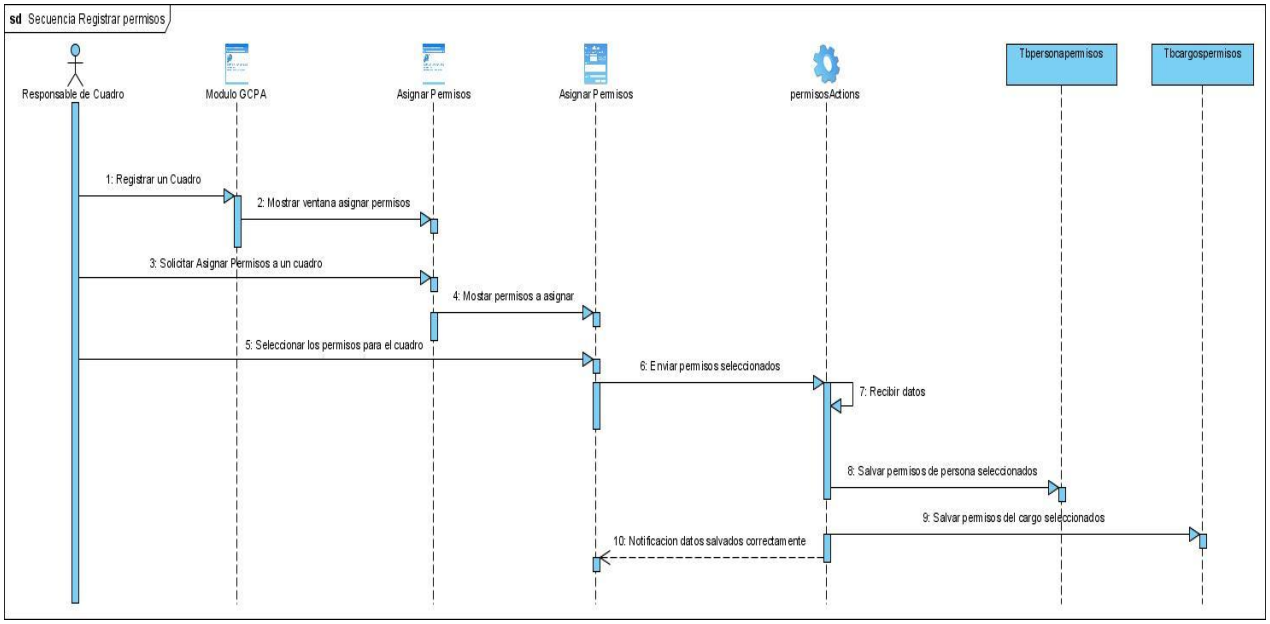

Figura 11. Secuencia: Registrar Permisos.

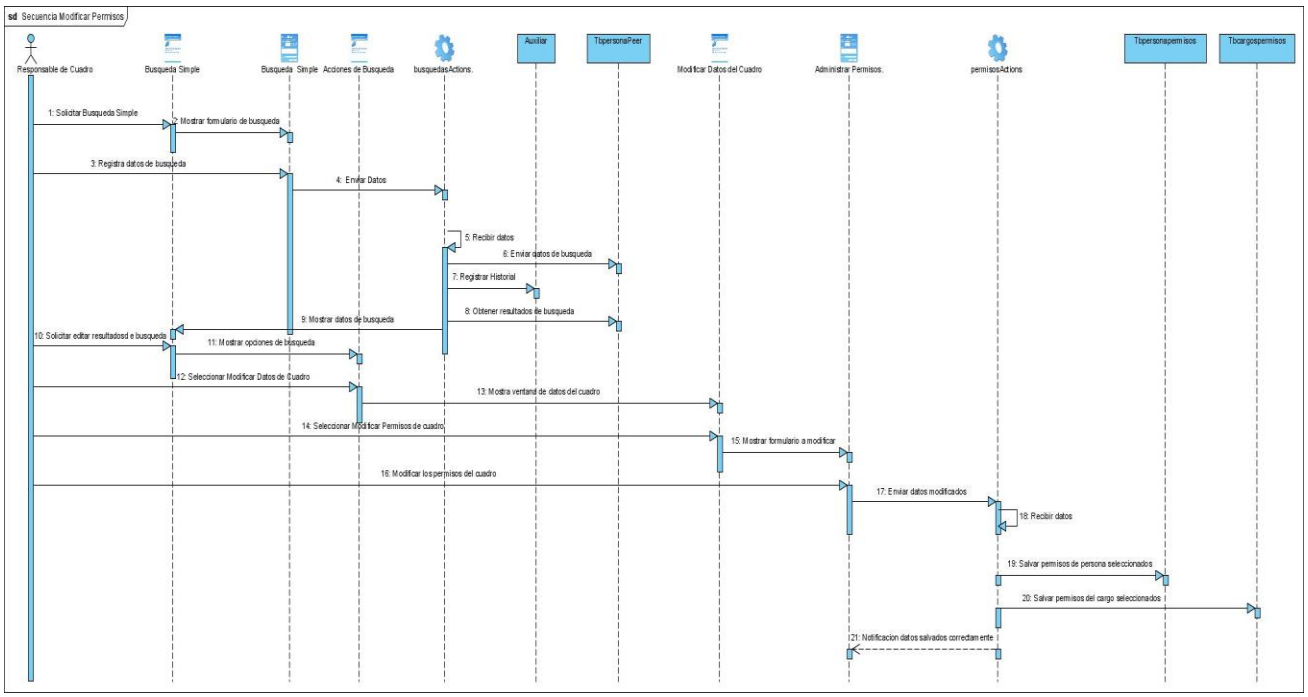

Figura 12. Secuencia: Modificar Permisos.

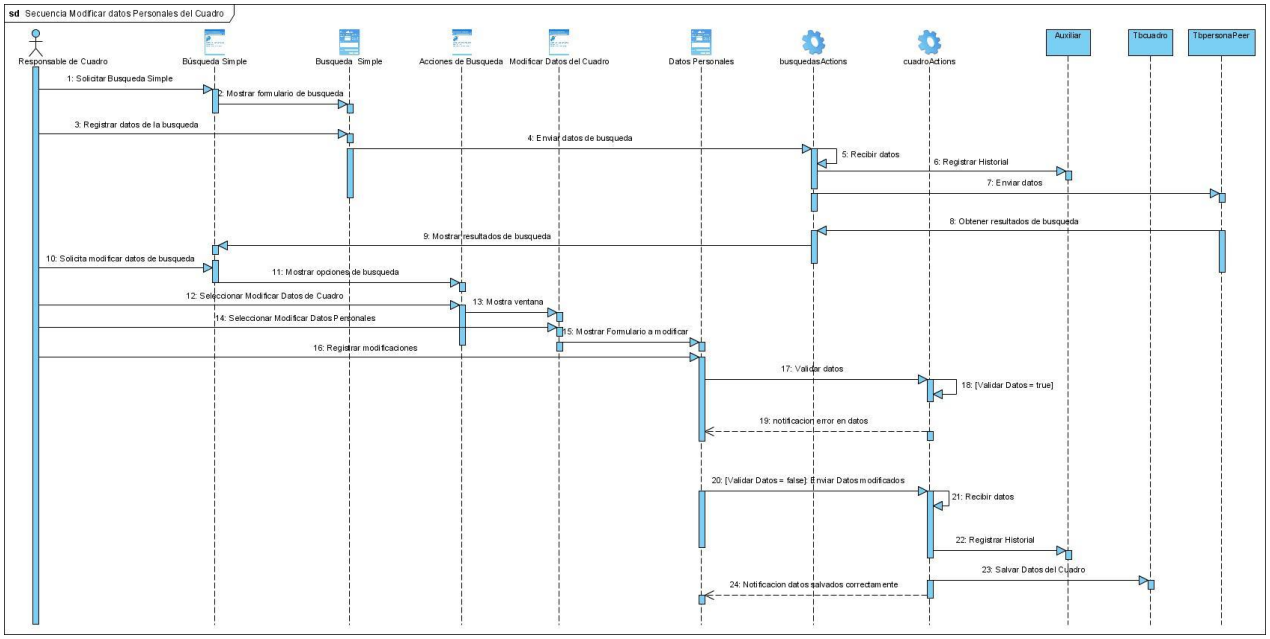

Figura 13. Secuencia: Modificar Datos Personales del Cuadro.

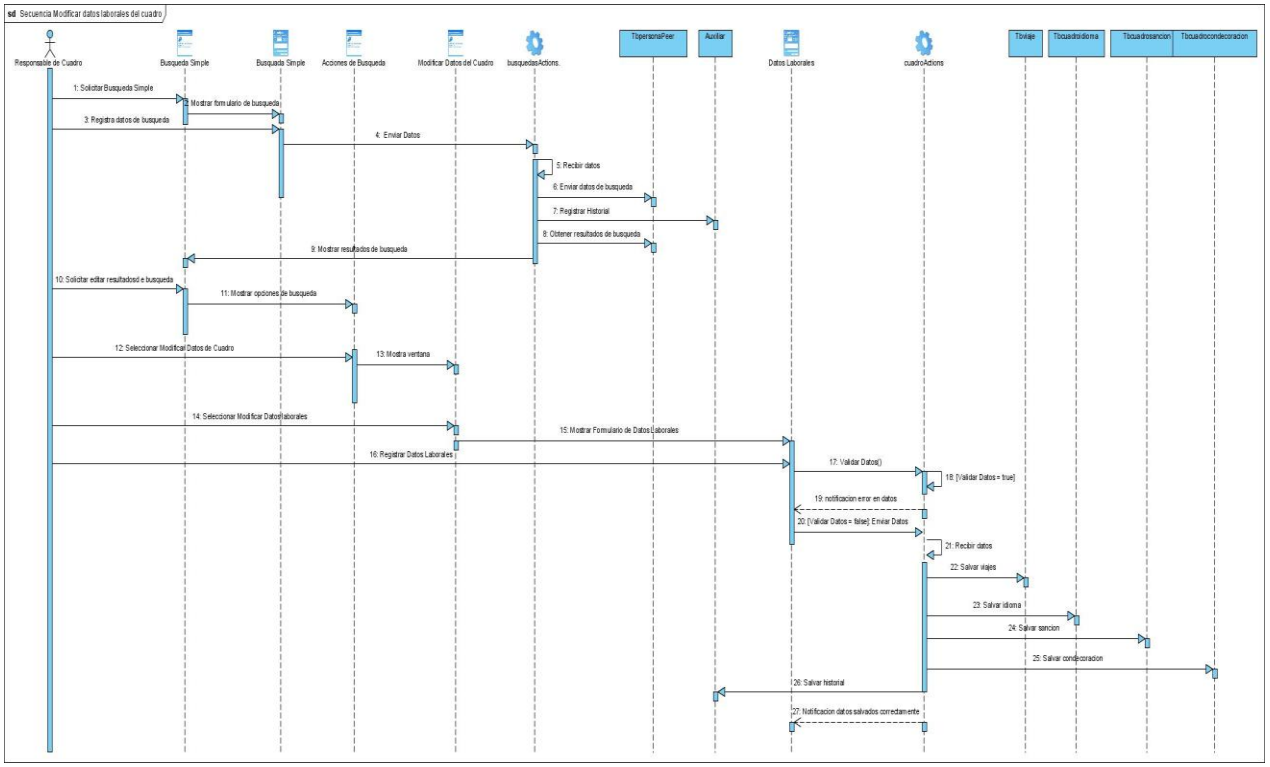

Figura 14. Secuencia: Modificar Datos Laborales del Cuadro.

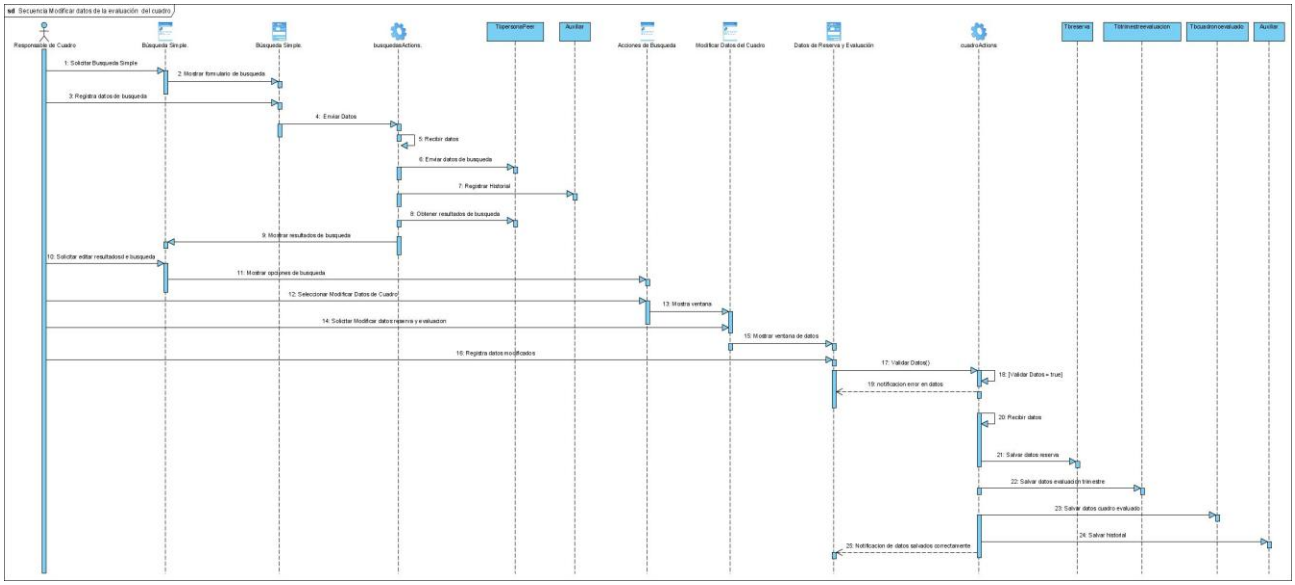

Figura 15. Secuencia: Modificar Datos de la Evaluación del Cuadro.

Anexo 2. Diagramas de Secuencia del Módulo de Personal de Apoyo.

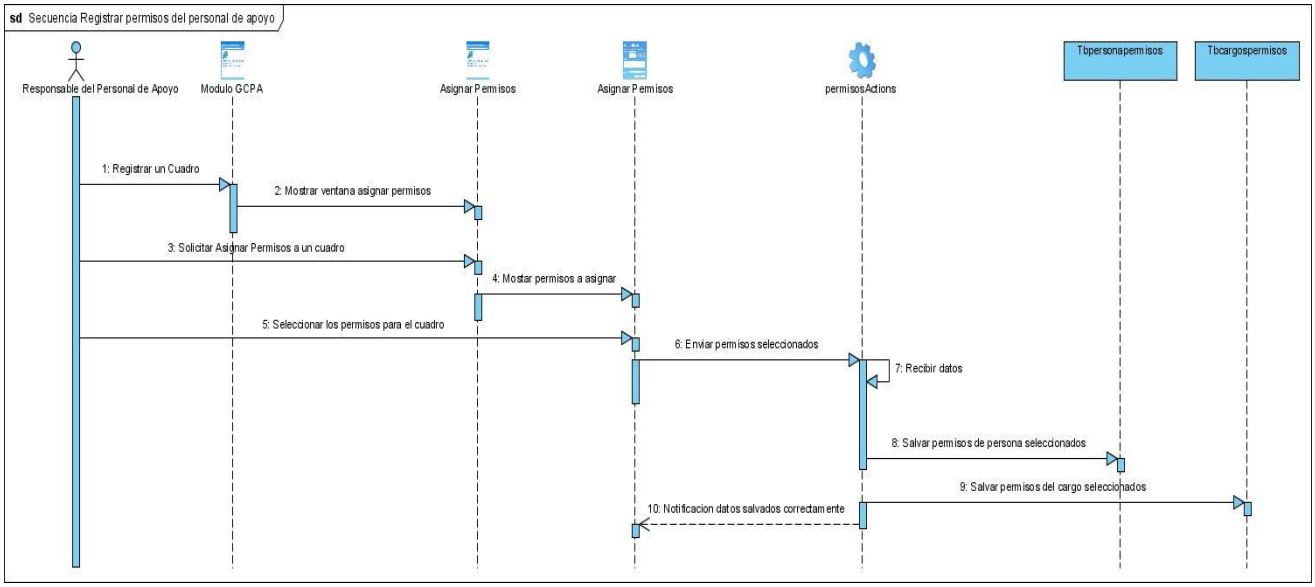

Figura 1. Secuencia: Registrar Permisos del Personal de Apoyo.

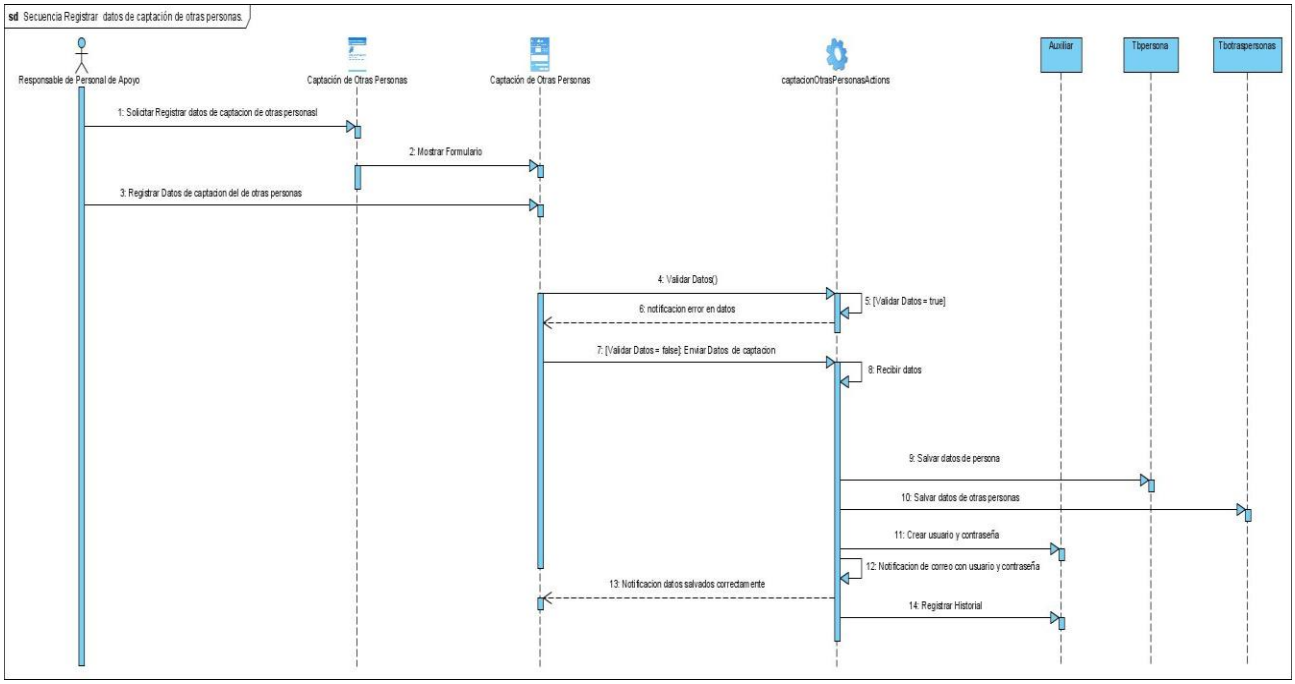

Figura 2. Secuencia: Registrar Datos de Captación de Otras Personas.

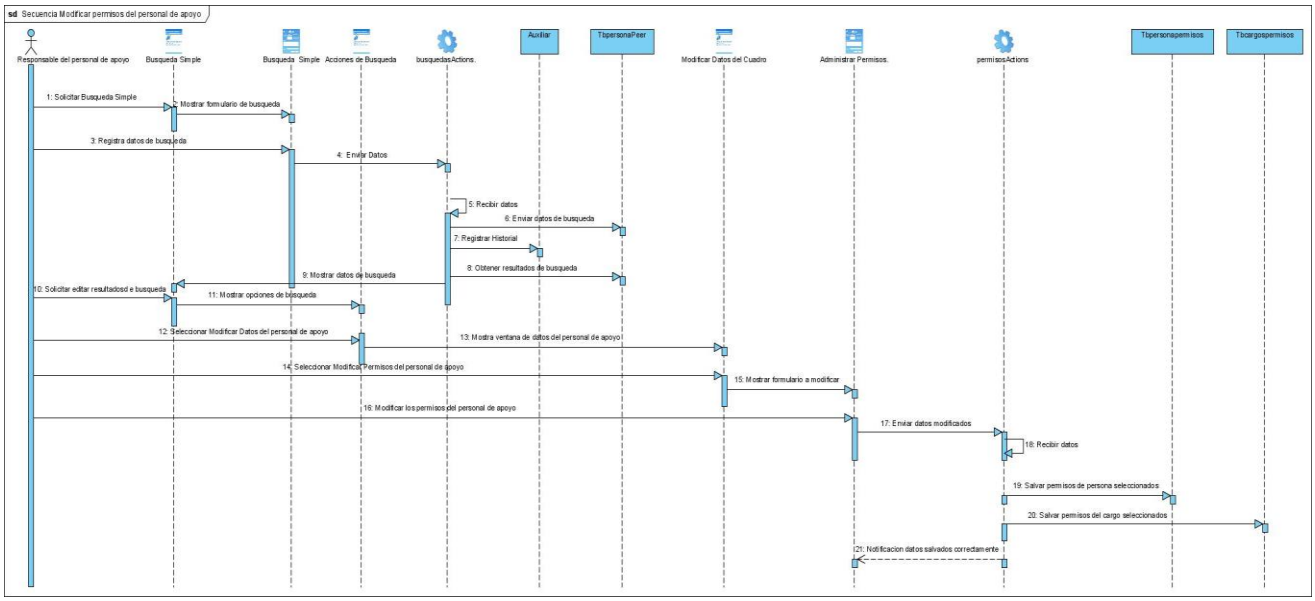

Figura 3. Secuencia: Modificar Permisos del Personal de Apoyo.

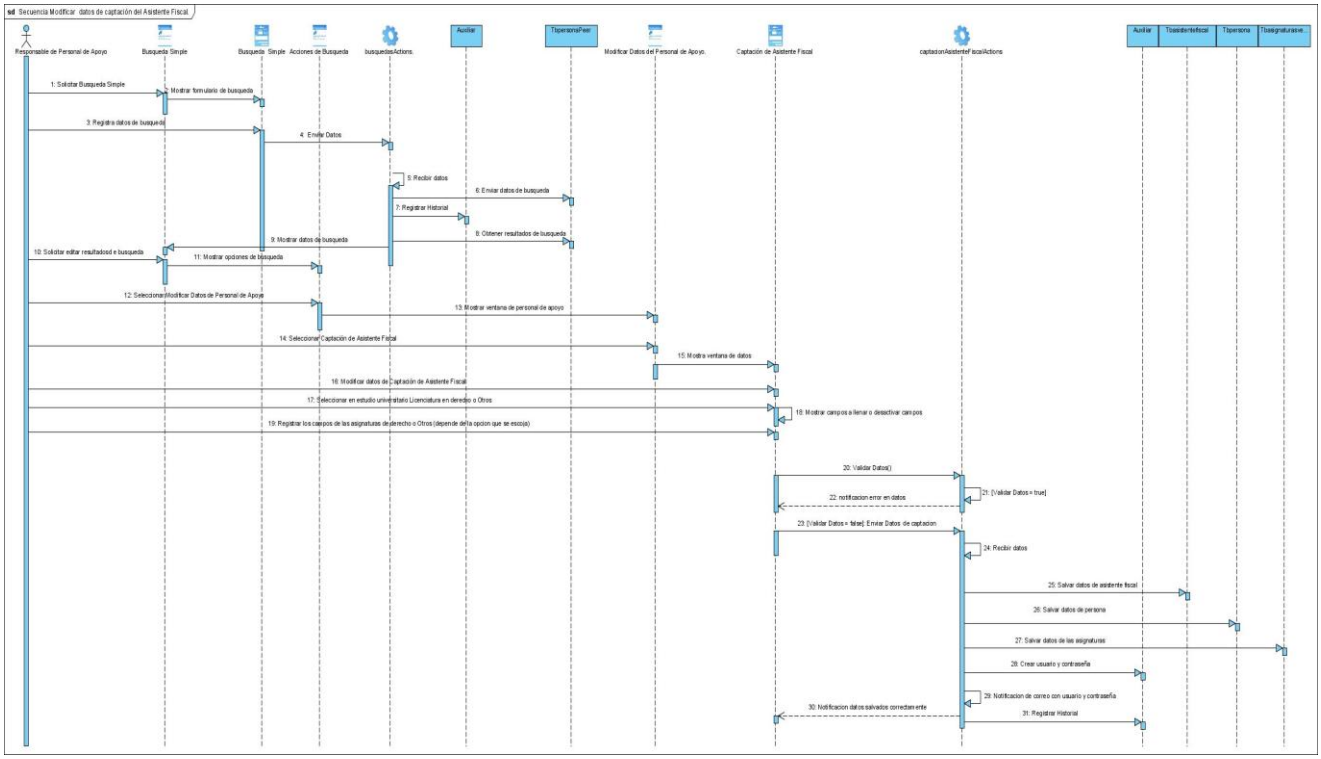

Figura 4. Secuencia: Modificar Datos de Captación del Asistente Fiscal.

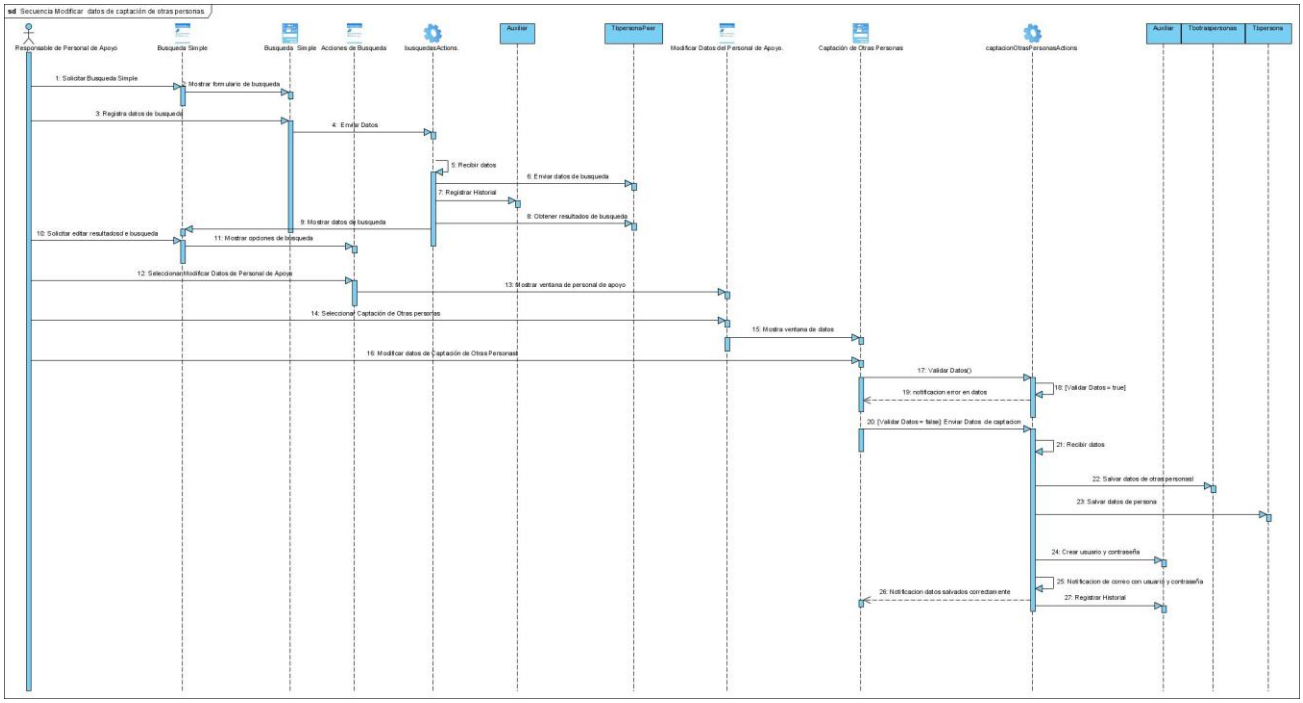

Figura 5. Secuencia: Modificar Datos de Captación de Otras Personas.

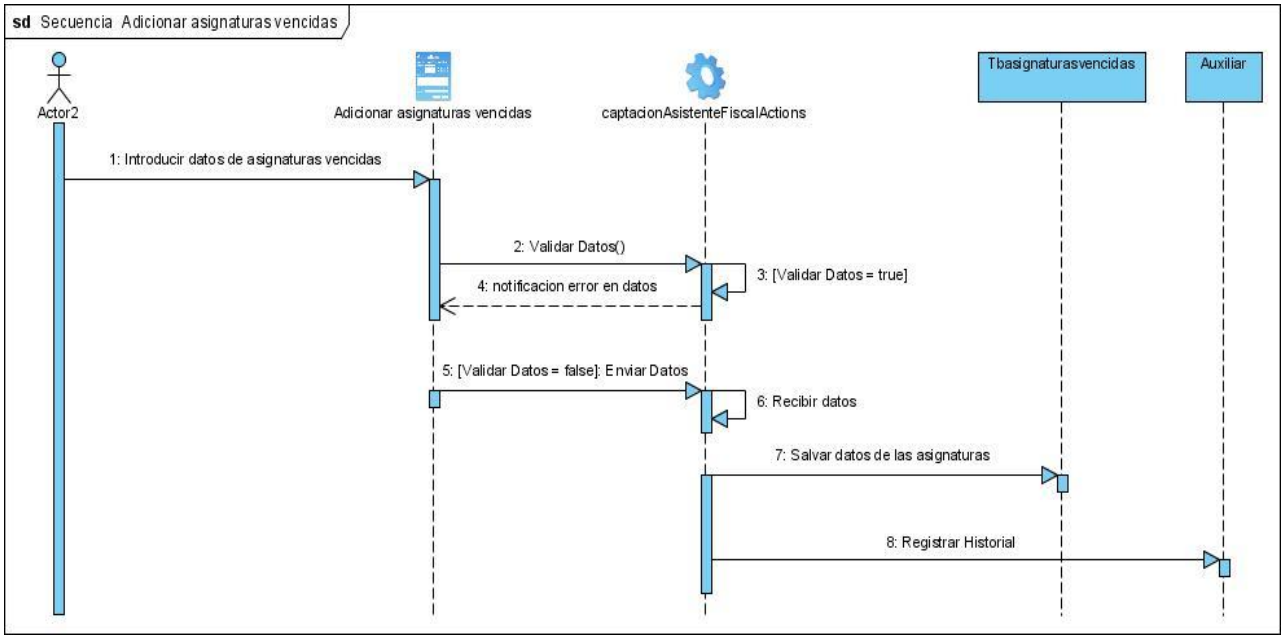

Figura 6. Secuencia: Adicionar Asignaturas Vencidas.

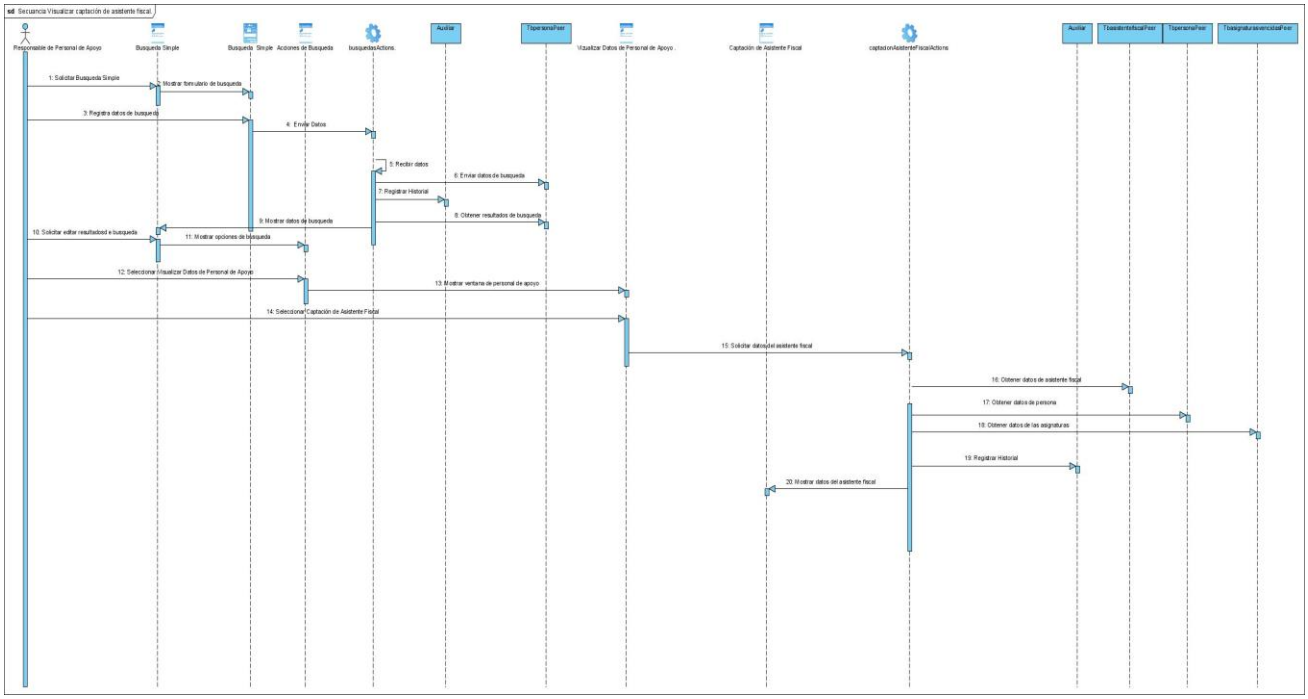

Figura 7. Secuencia: Visualizar Captación de Asistente Fiscal.

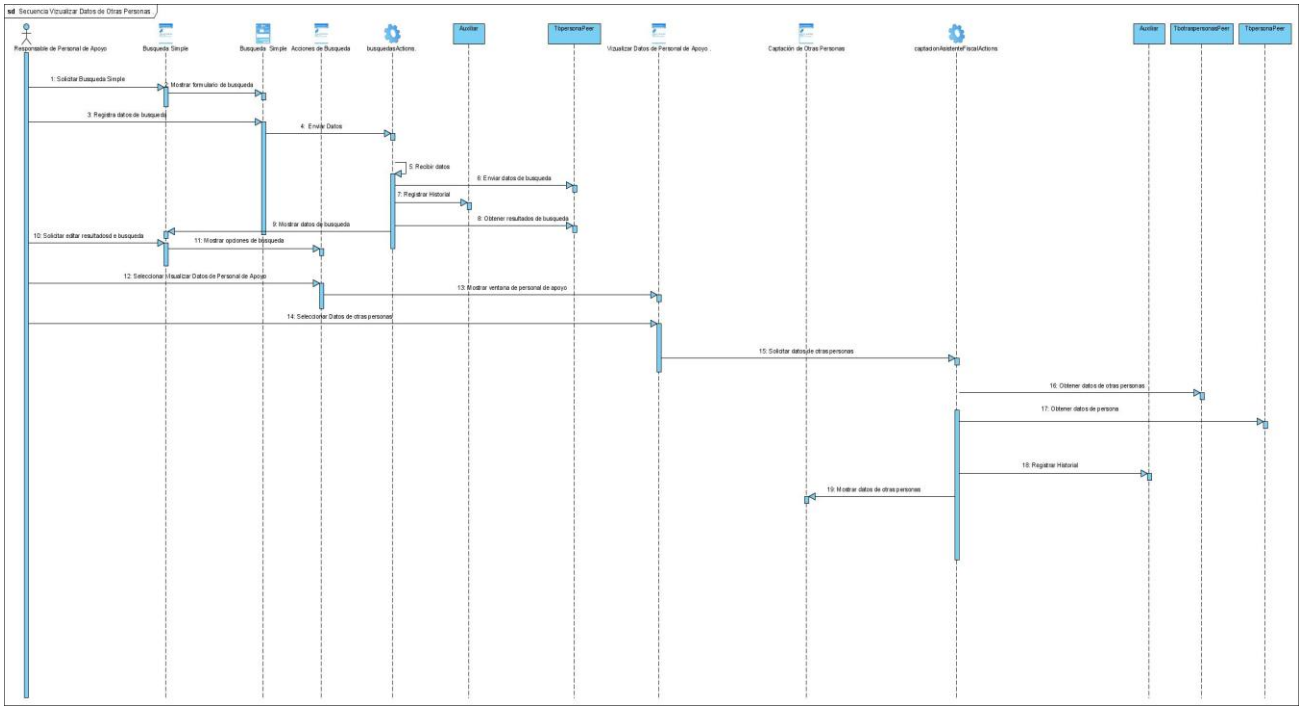

Figura 8. Secuencia: Visualizar Captación de Otras Personas.

Anexo 3. Diagramas de Secuencia del Módulo Capacitación.

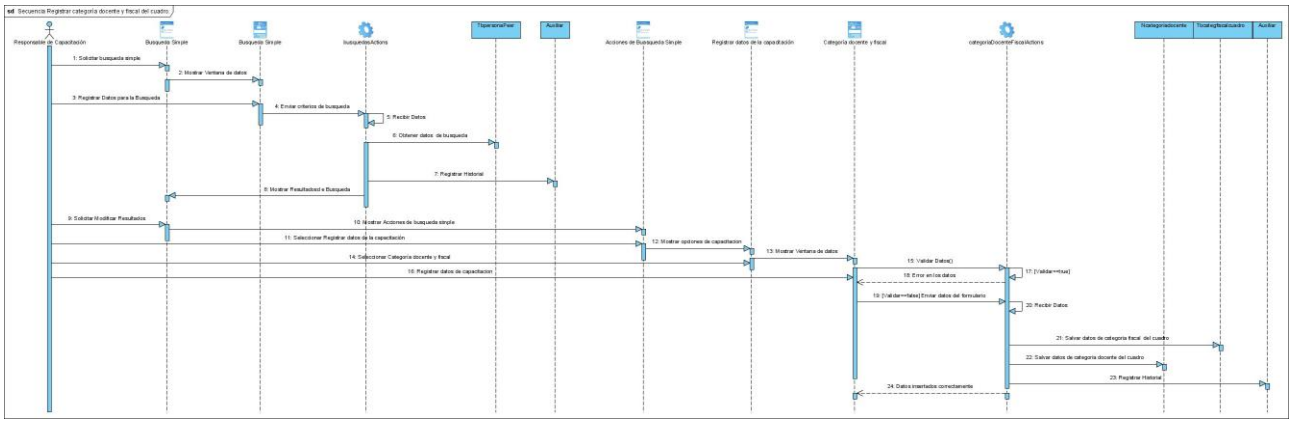

Figura 1. Secuencia: Registrar Categoría Docente y Fiscal del Cuadro.

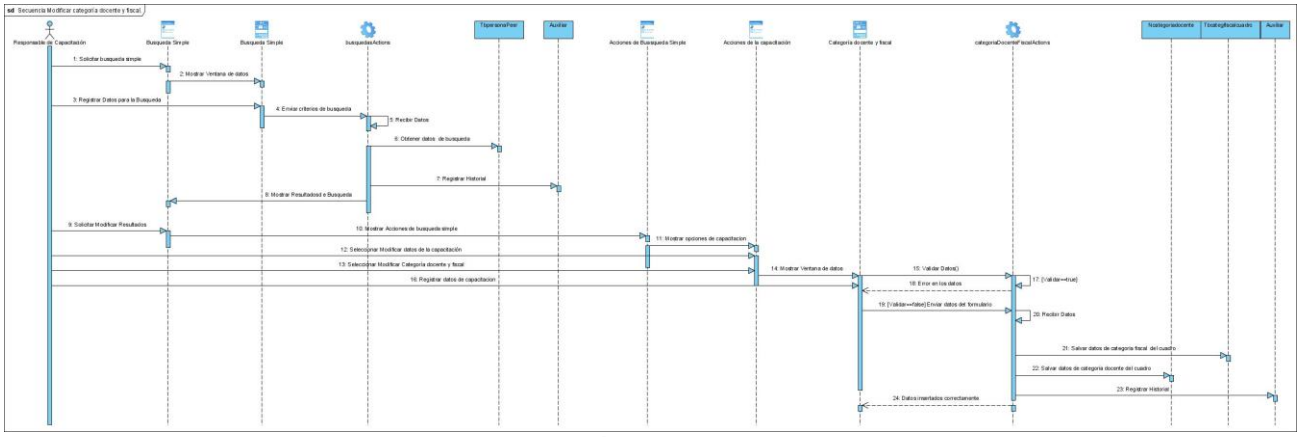

Figura 2. Secuencia: Modificar Categoría Docente y Fiscal.

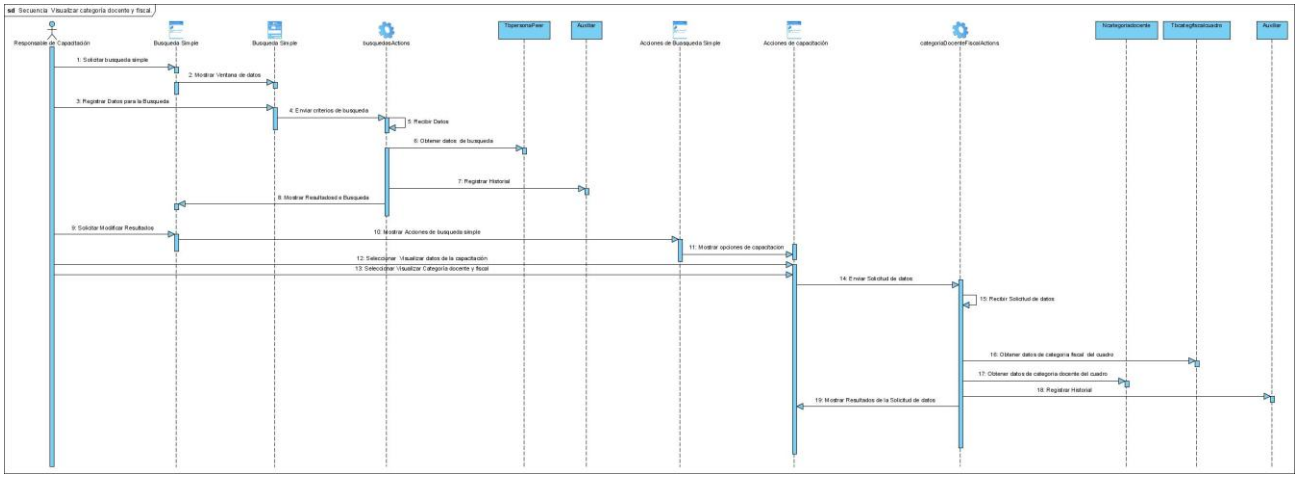

Figura 3. Secuencia: Visualizar Categoría Docente y Fiscal.

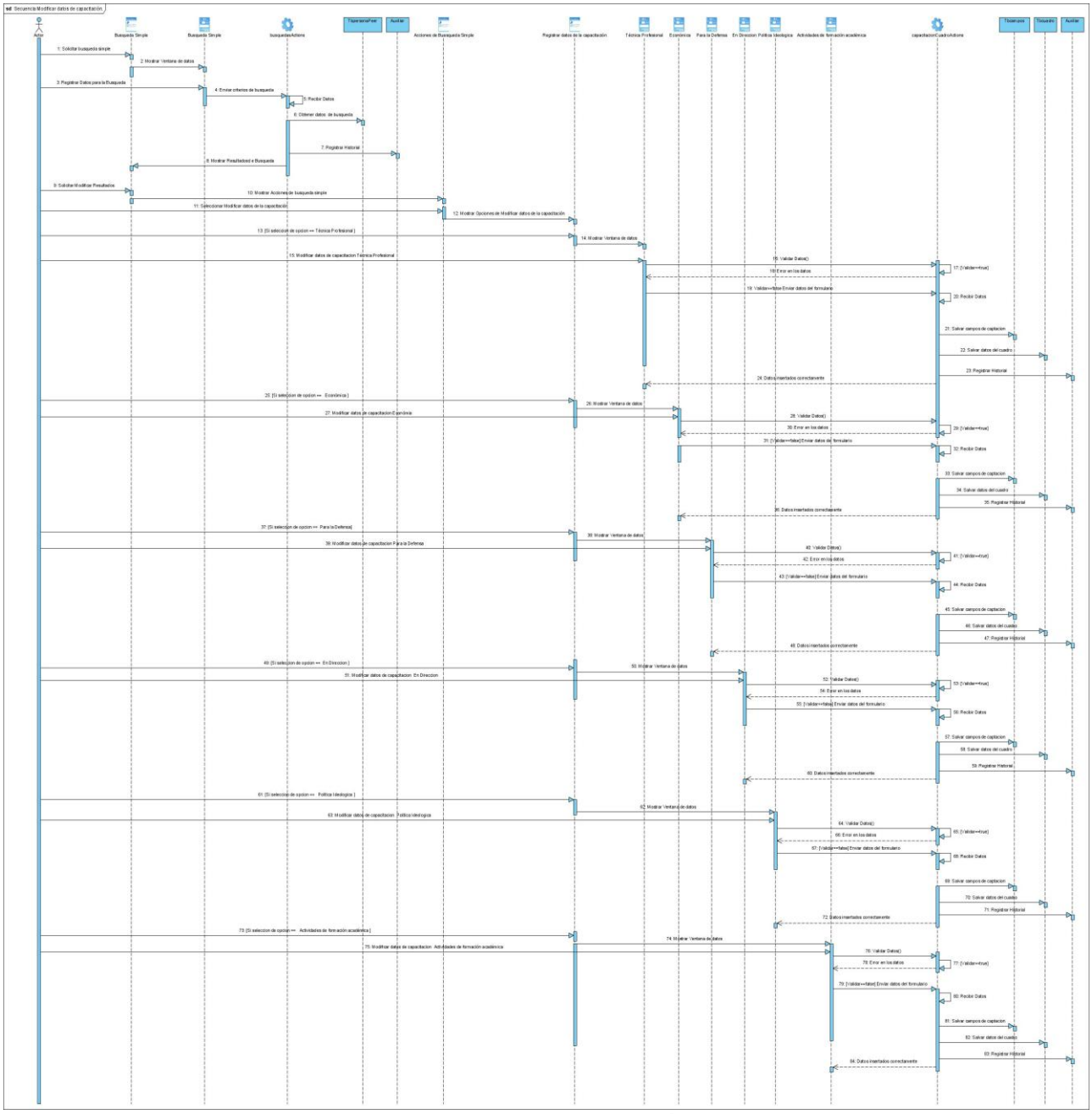

Figura 4. Secuencia: Modificar Datos de Capacitación.

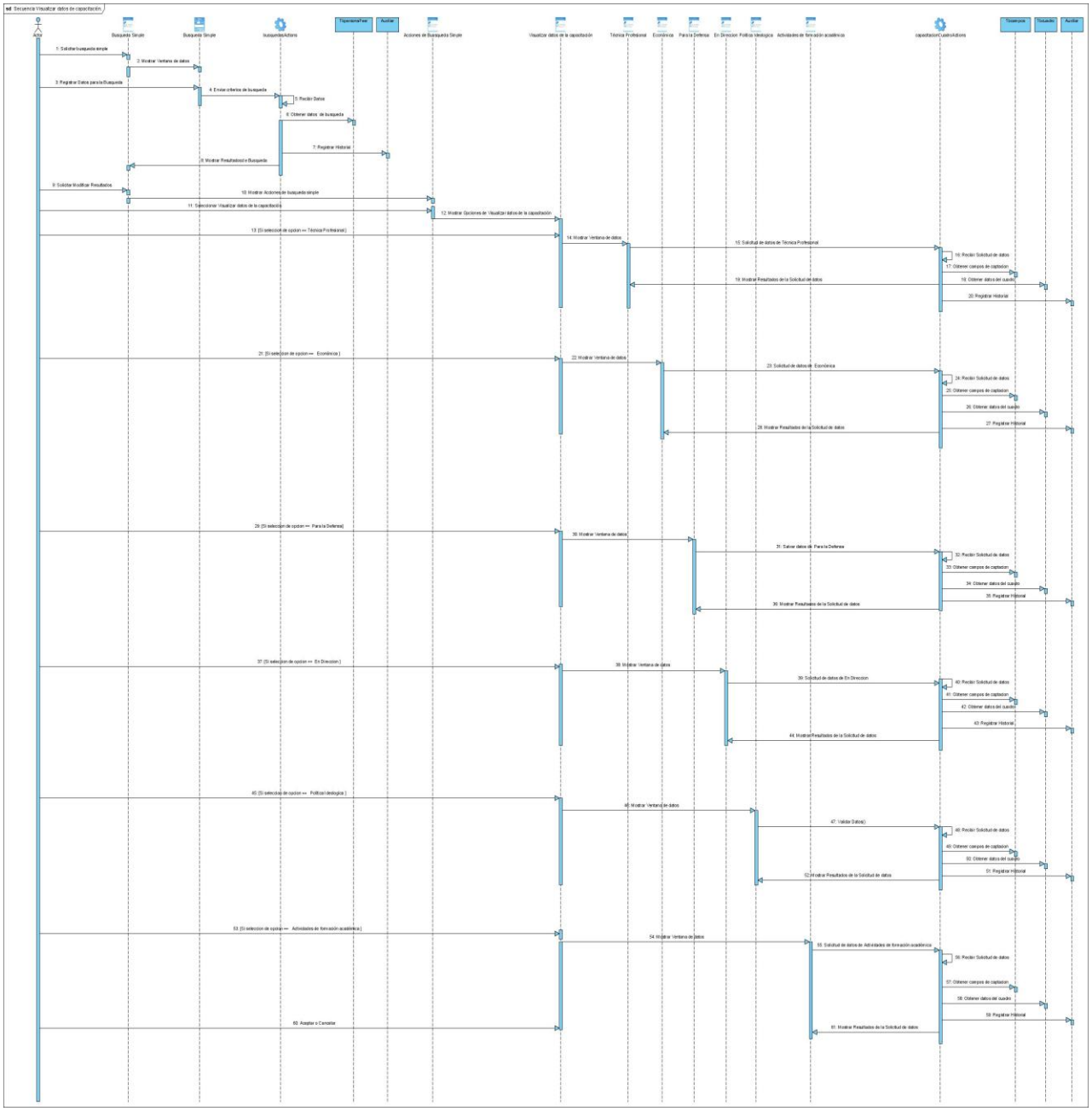

Figura 5. Secuencia: Visualizar Datos de Capacitación.

Anexo 4. Diagramas de Secuencia del Módulo Captación de Plantilla.

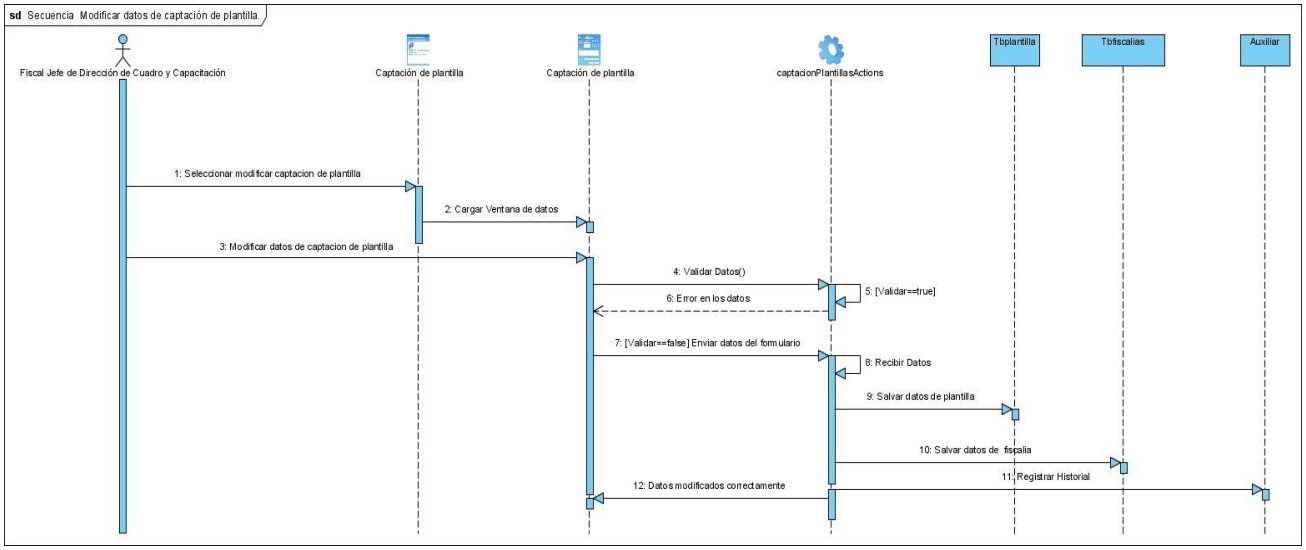

Figura 1. Secuencia: Modificar Datos de Captación de Plantilla.

Anexo 5. Diagramas de Secuencia del Módulo Generalidades.

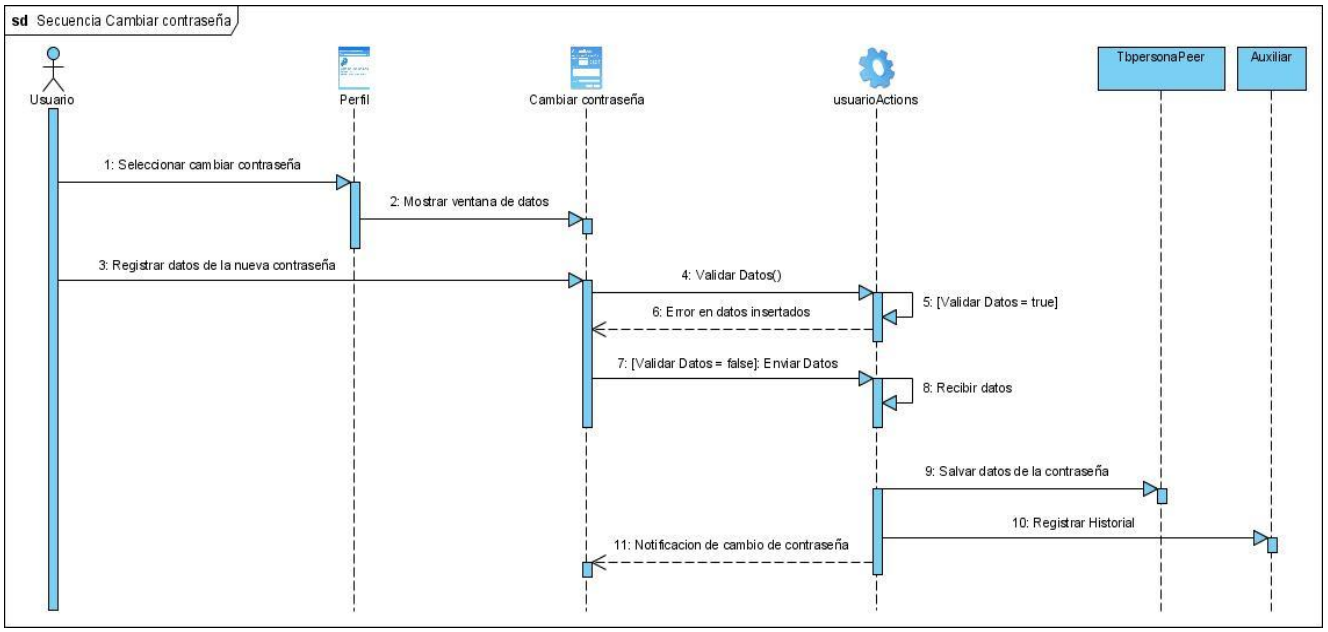

Figura 1. Secuencia: Cambiar Contraseña.

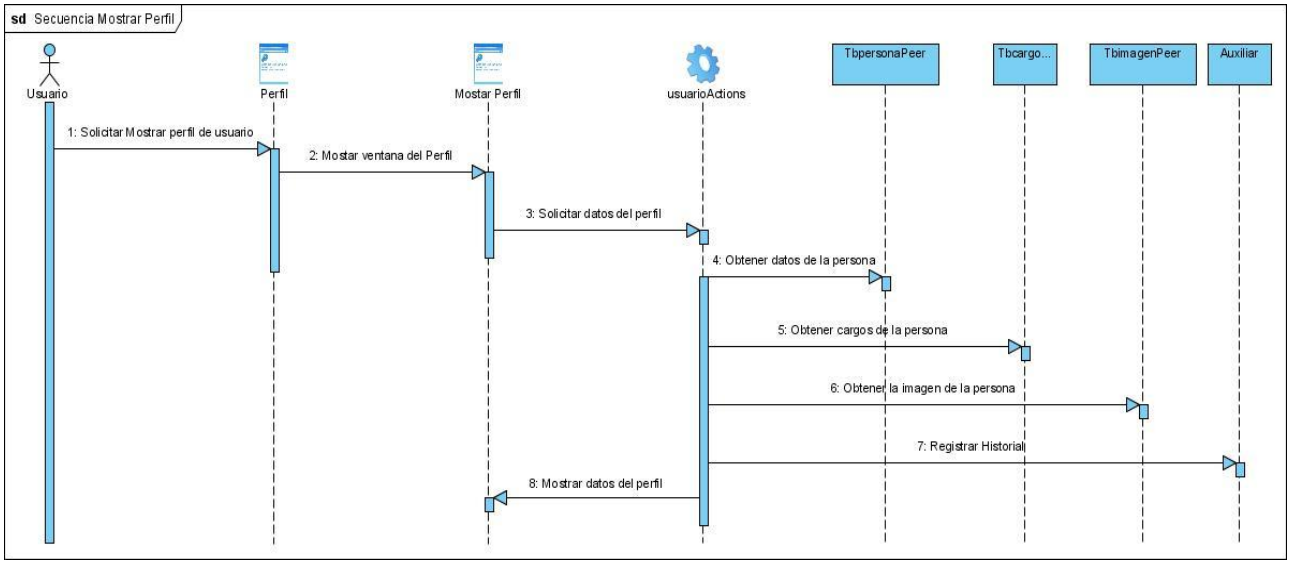

Figura 2. Secuencia: Mostrar Perfil.

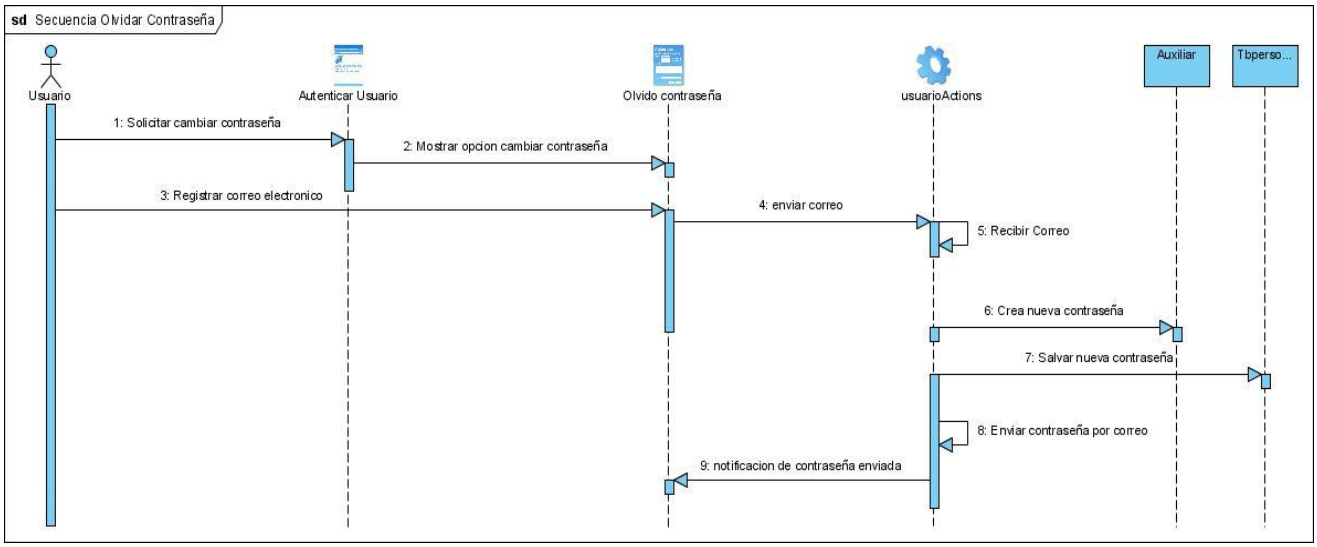

Figura 3. Secuencia: Olvidar Contraseña.

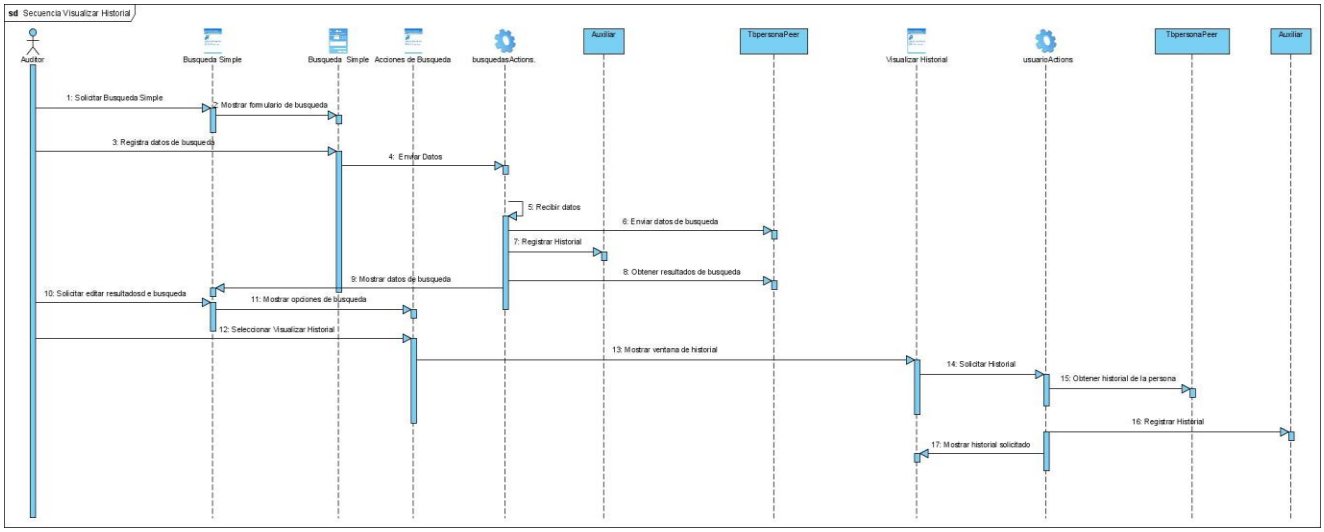

Figura 4. Secuencia: Visualizar Historial.

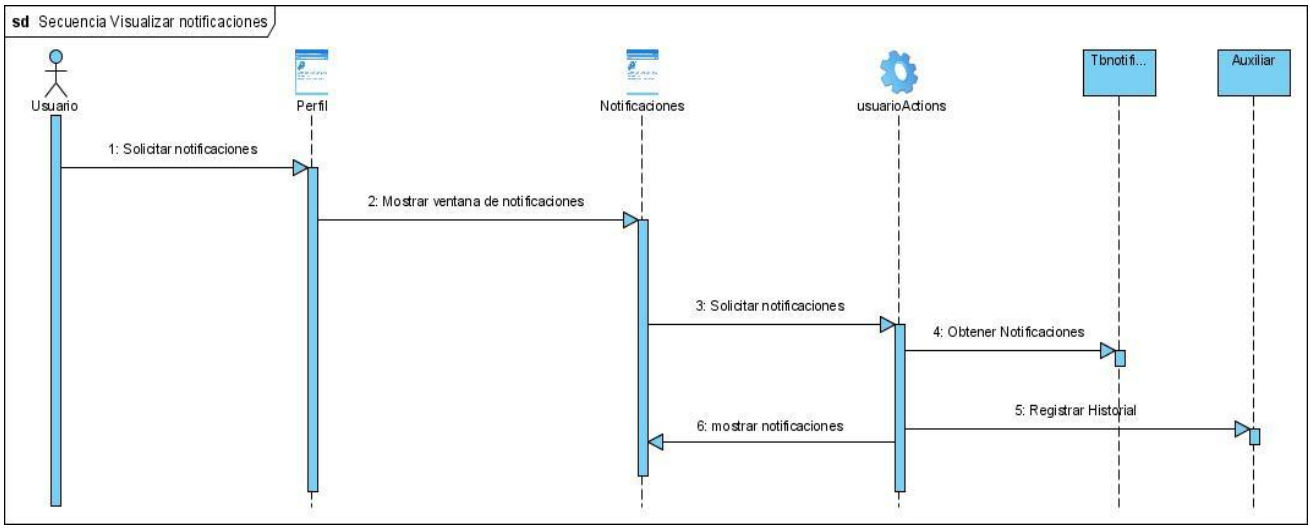

Figura 5. Secuencia: Visualizar Notificaciones.

Anexo 6. Diagramas de Secuencia del Módulo de Reporte y Búsqueda.

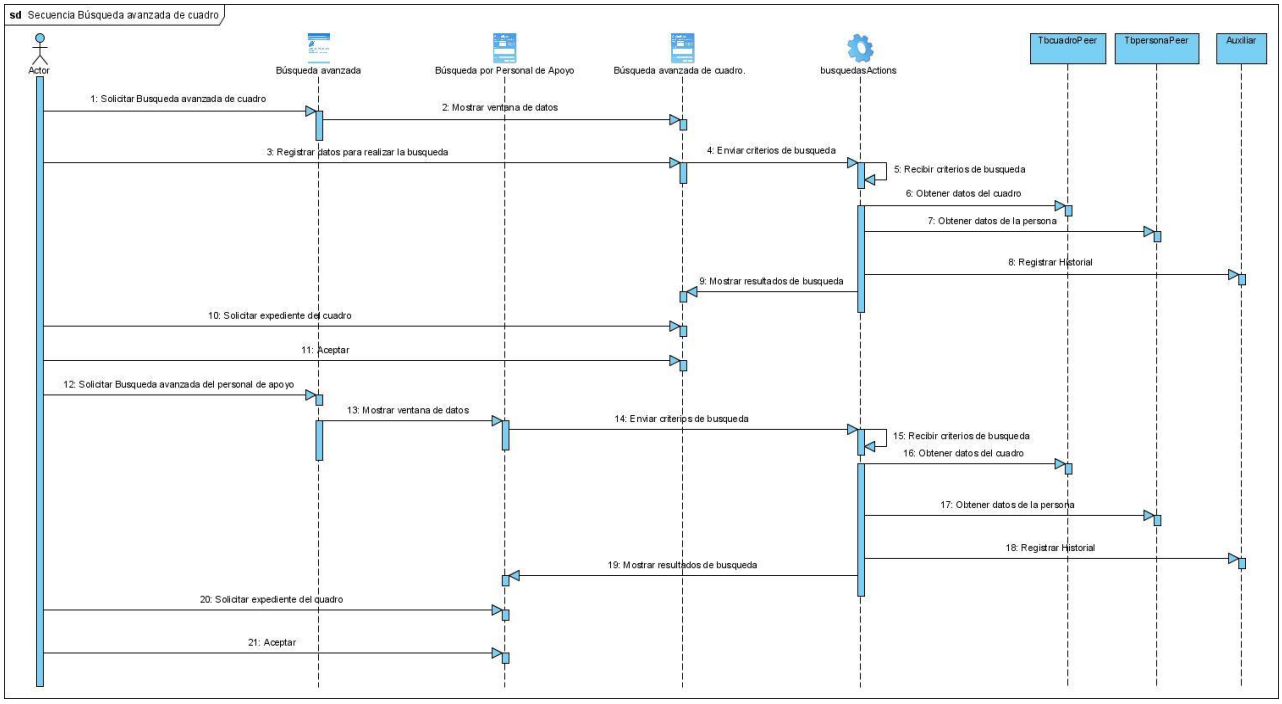

Figura 1. Secuencia: Búsqueda Avanzada de Cuadro.

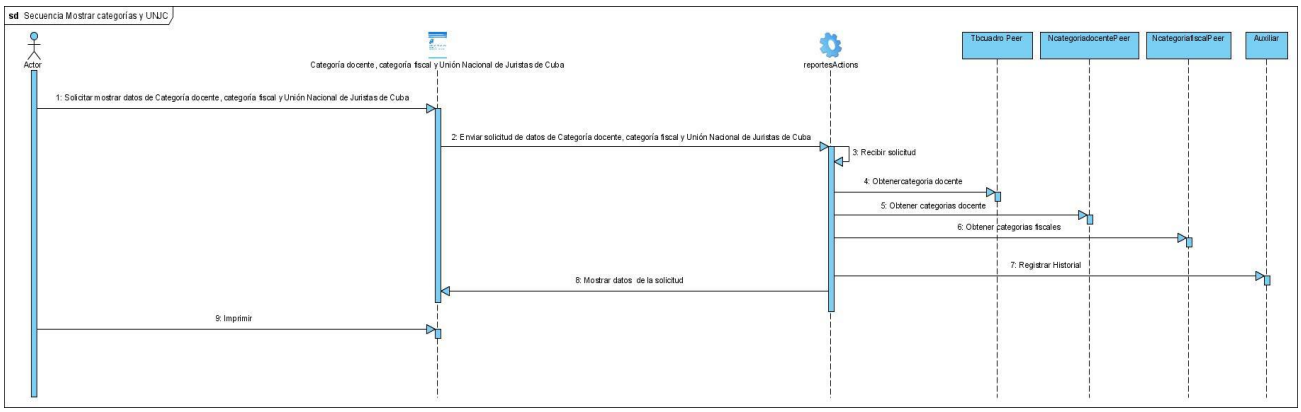

Figura 2. Secuencia: Categoría Docente, Categoría Fiscal y Unión Nacional de Juristas de Cuba.

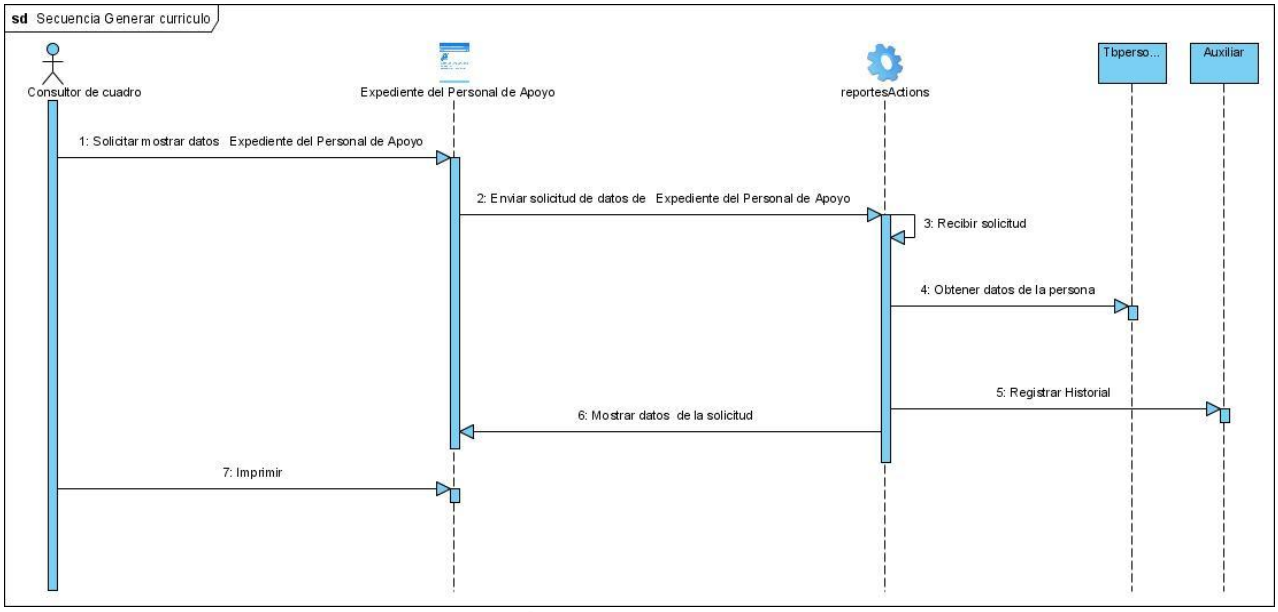

Figura 3. Secuencia: Generar Currículo del Cuadro.

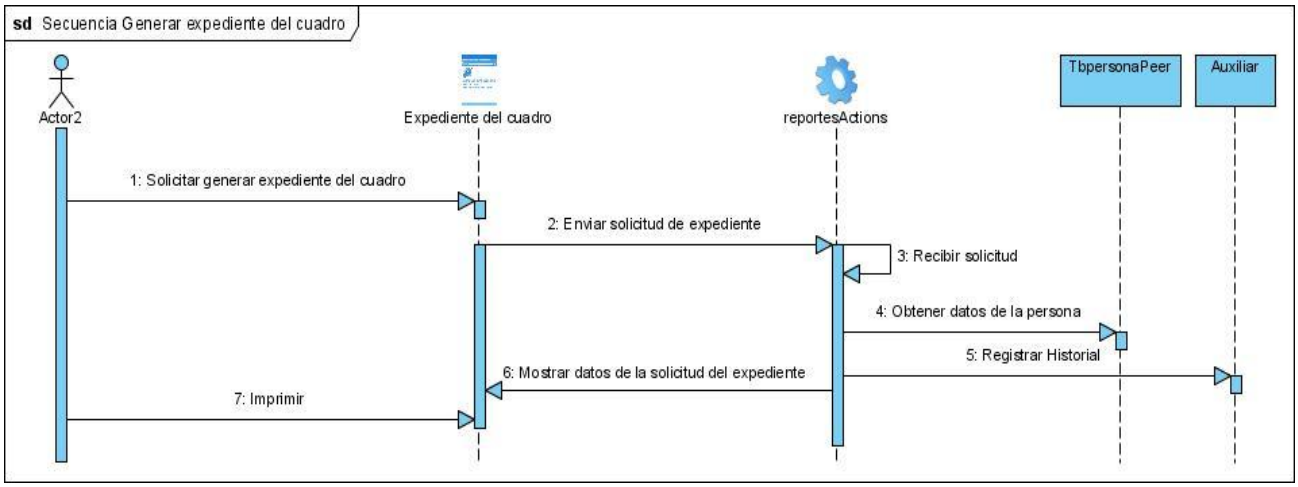

Figura 4. Secuencia: Generar Expediente del Cuadro.

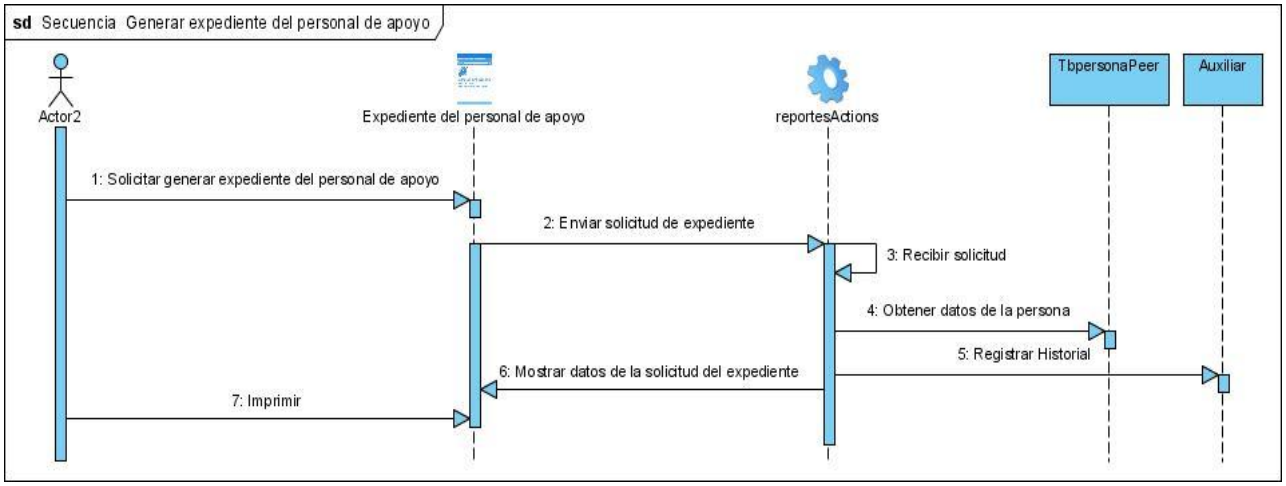

Figura 5. Secuencia: Generar Expediente del Personal de Apoyo.

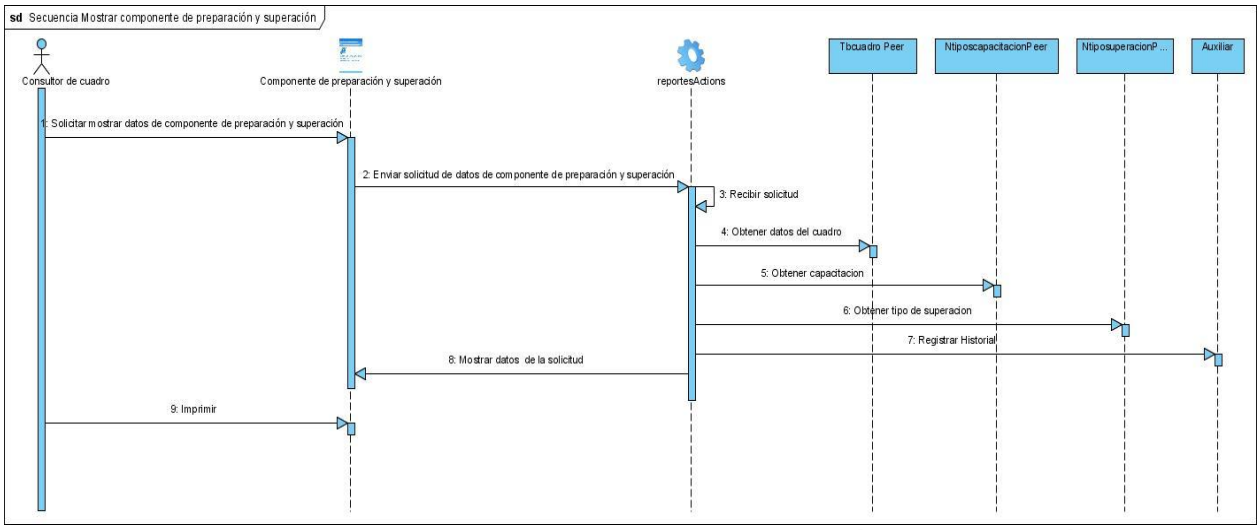

Figura 6. Secuencia: Mostrar Componente de Preparación y Superación.

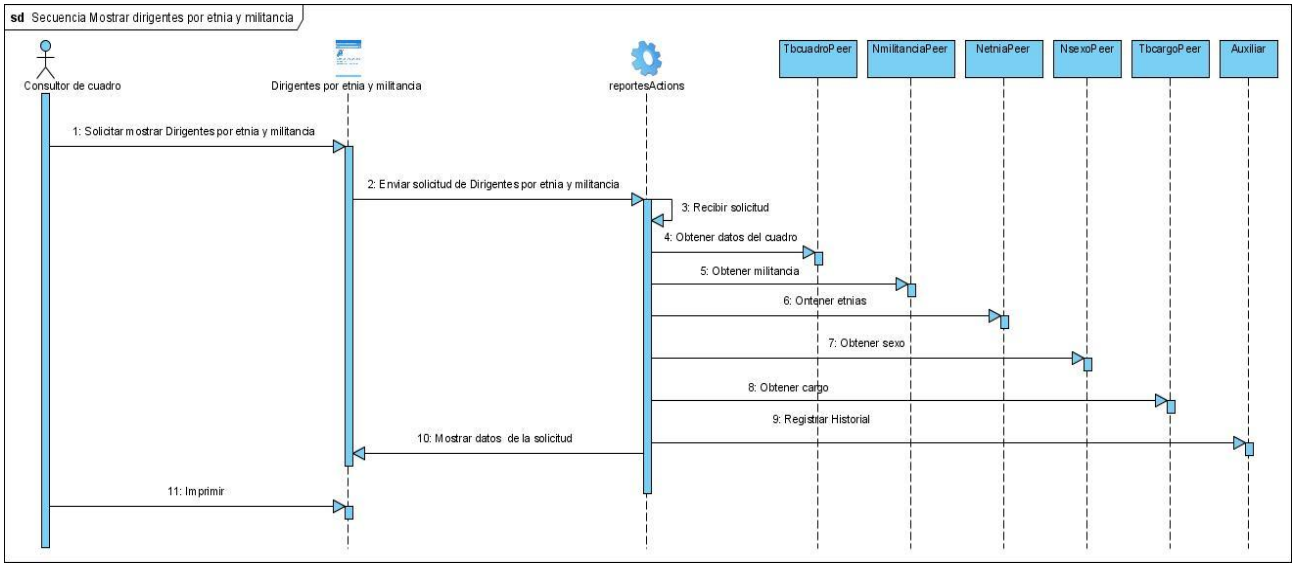

Figura 7. Secuencia: Mostrar Dirigentes por Etnia y Militancia.

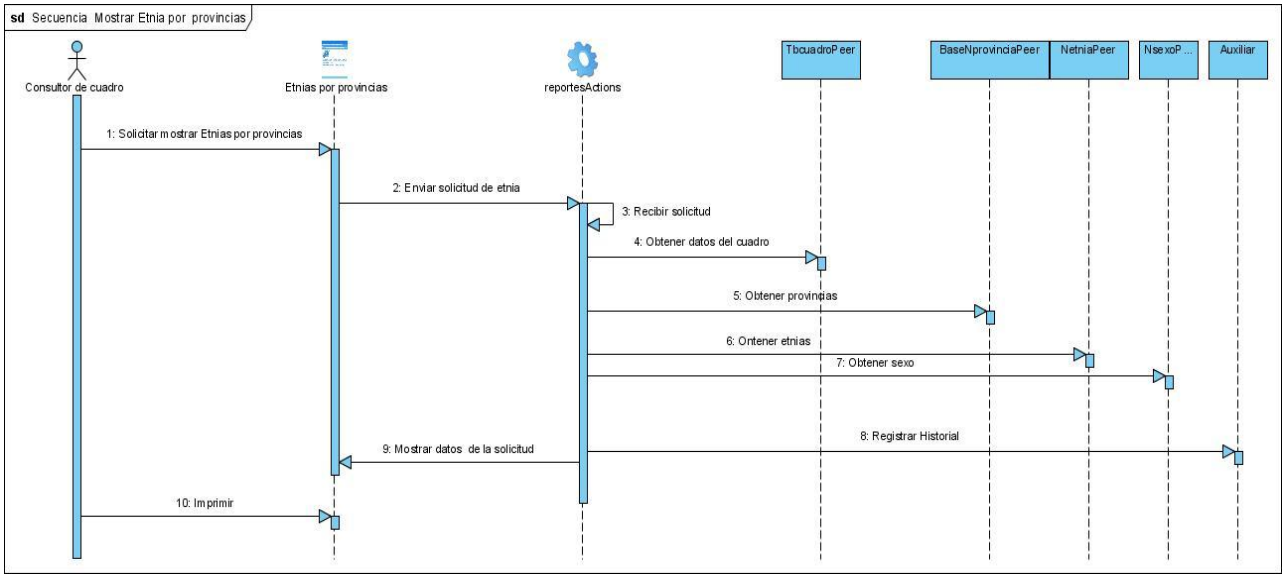

Figura 8. Secuencia: Mostrar Etnia por Provincias.

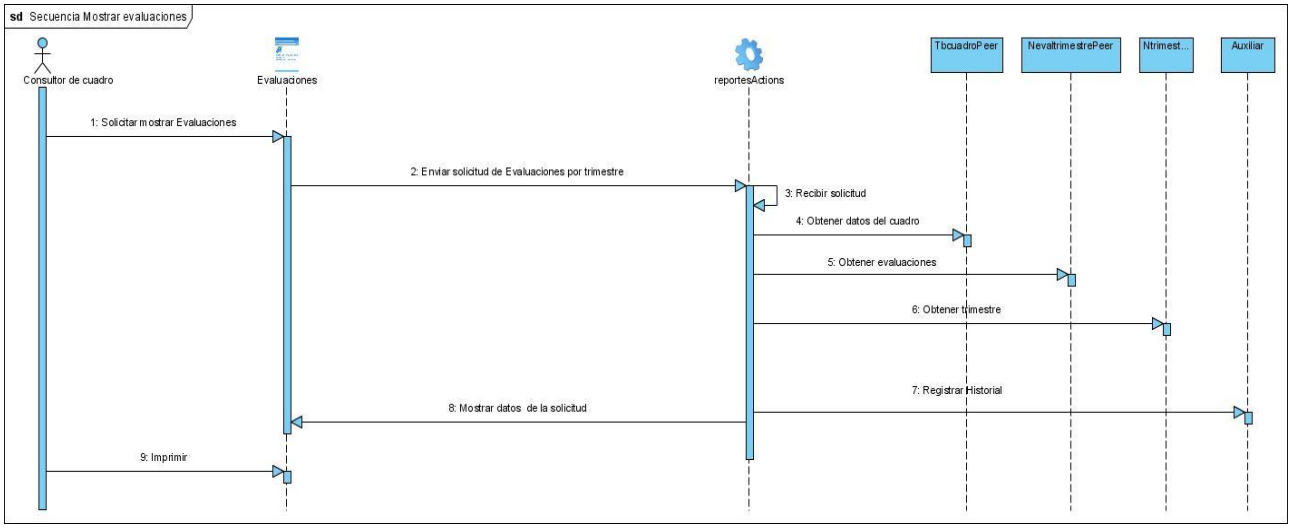

Figura 9. Secuencia: Mostrar Evaluaciones.

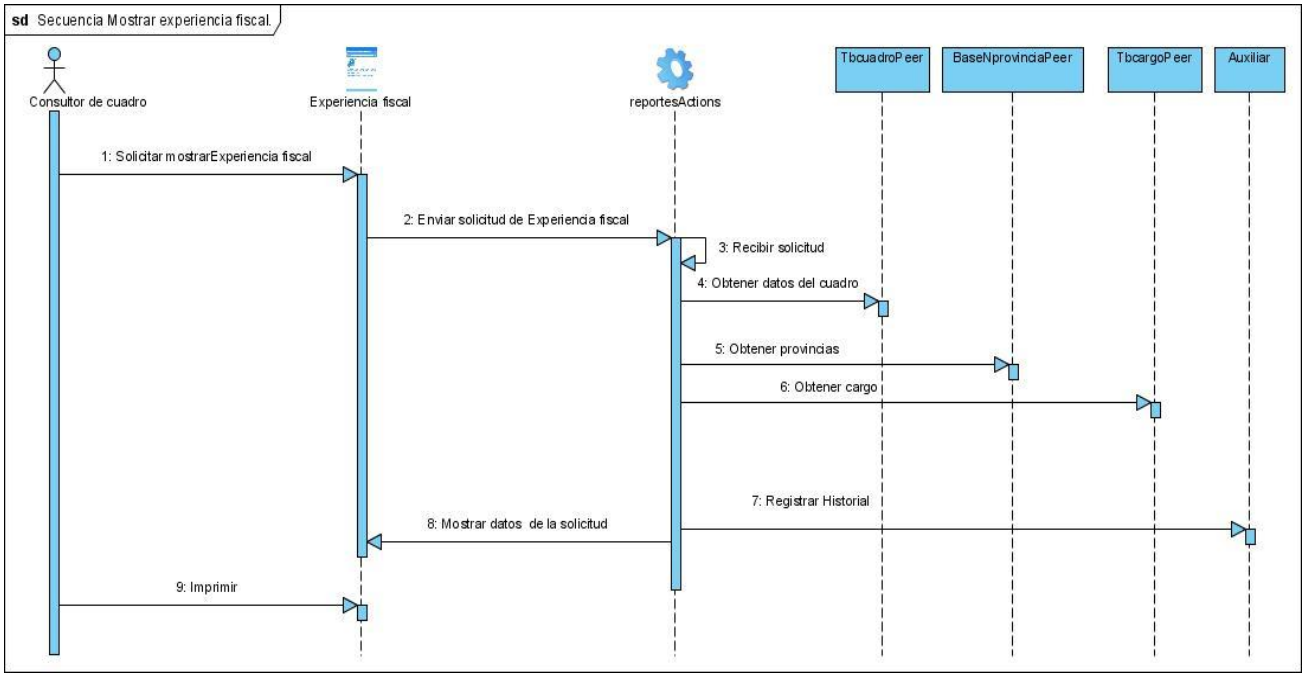

Figura 10. Secuencia: Mostrar Experiencia Fiscal.

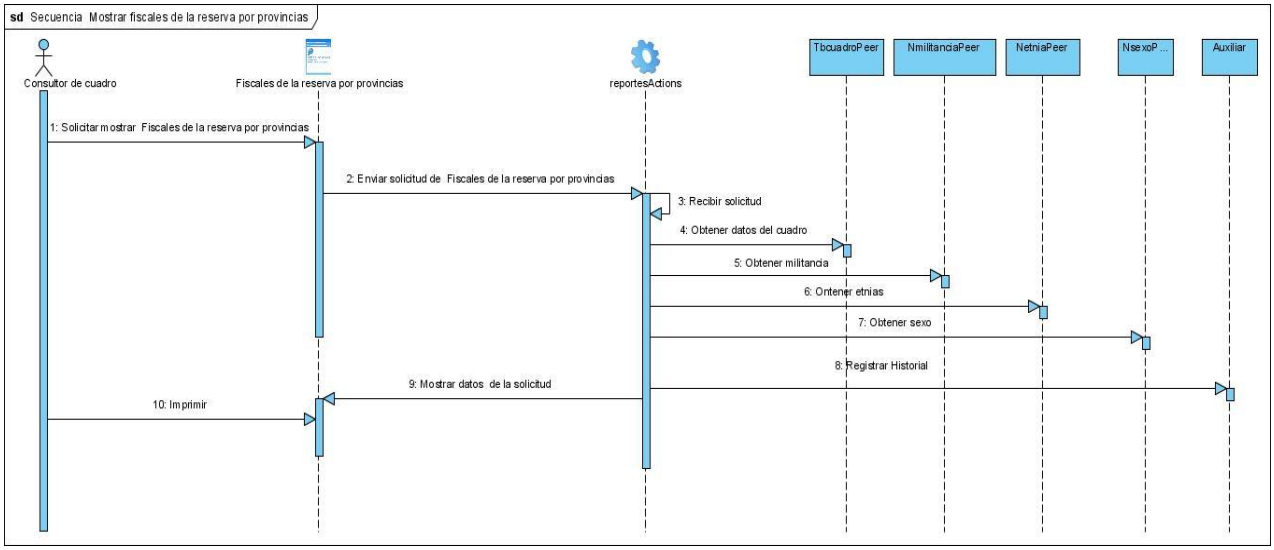

Figura 11. Secuencia: Mostrar Fiscales de la Reserva por Provincias.

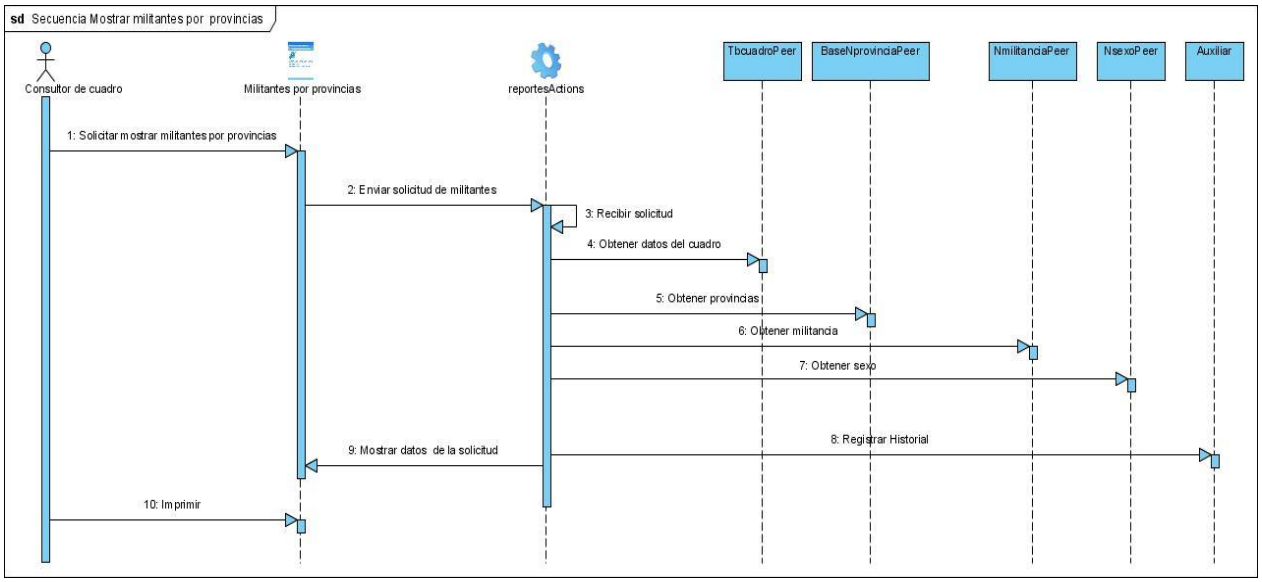

Figura 12. Secuencia: Mostrar Militantes por Provincias.

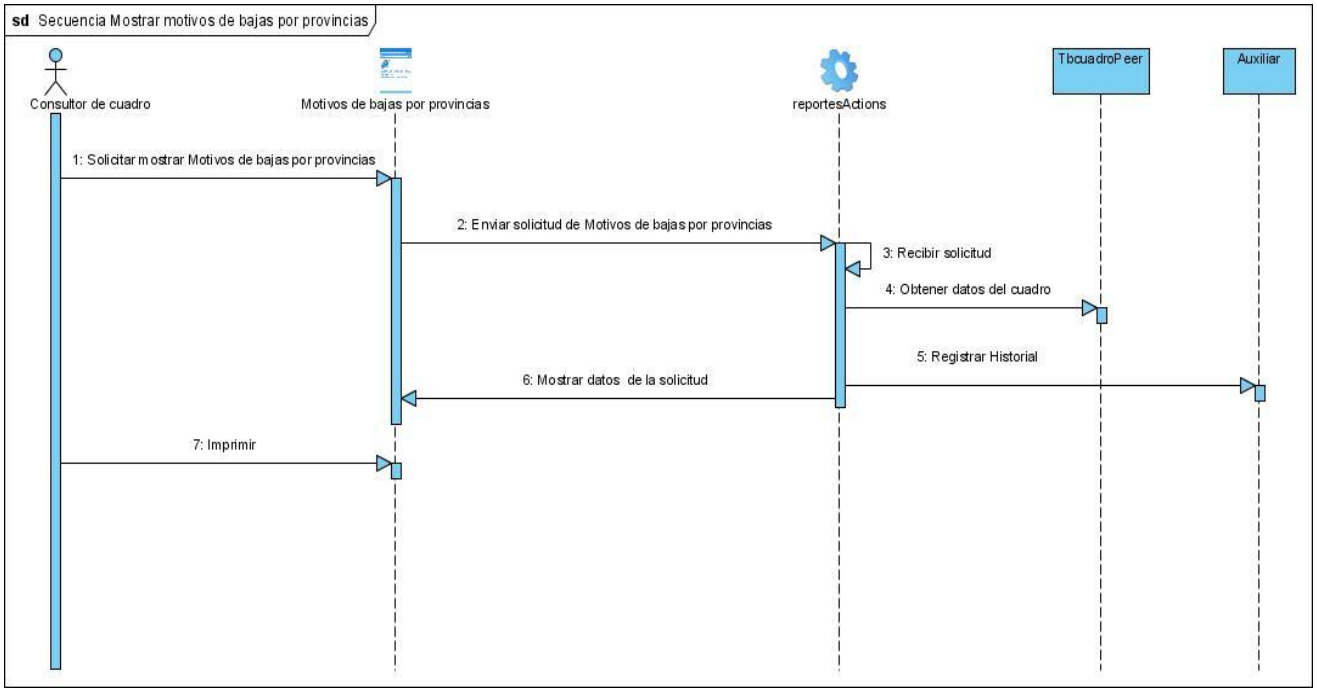

Figura 13. Secuencia: Mostrar Motivos de Bajas por Provincias.

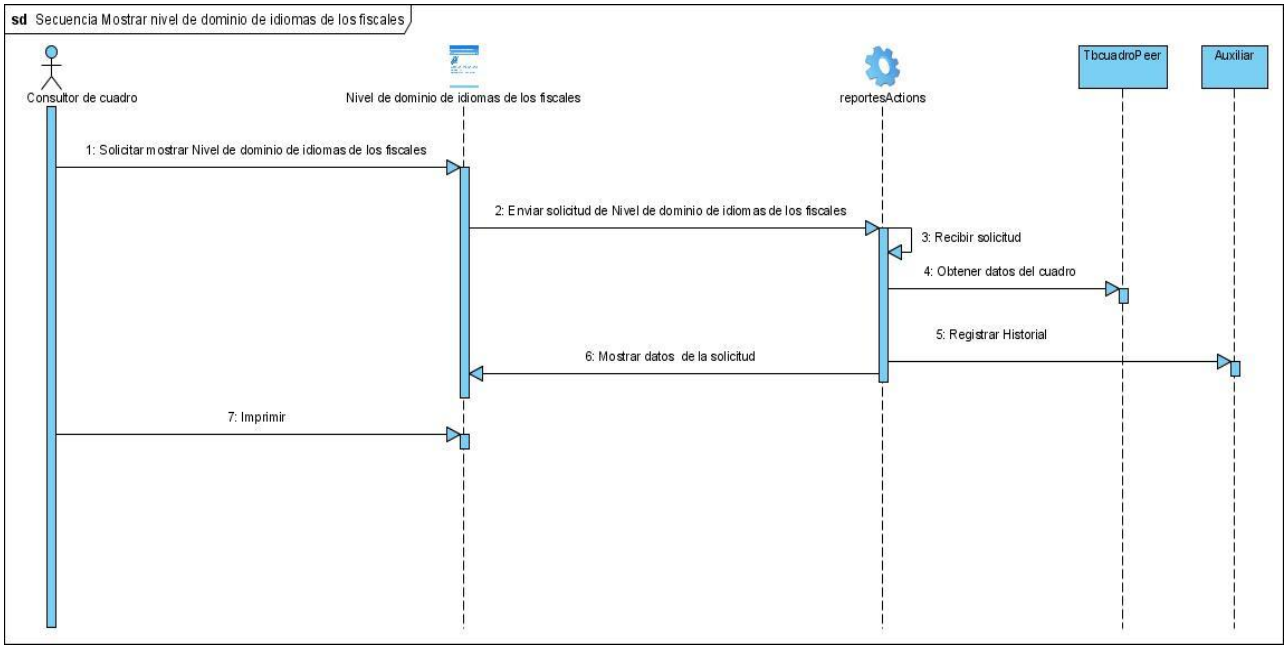

Figura 14. Secuencia: Mostrar Nivel de Dominio de Idiomas de los Fiscales.

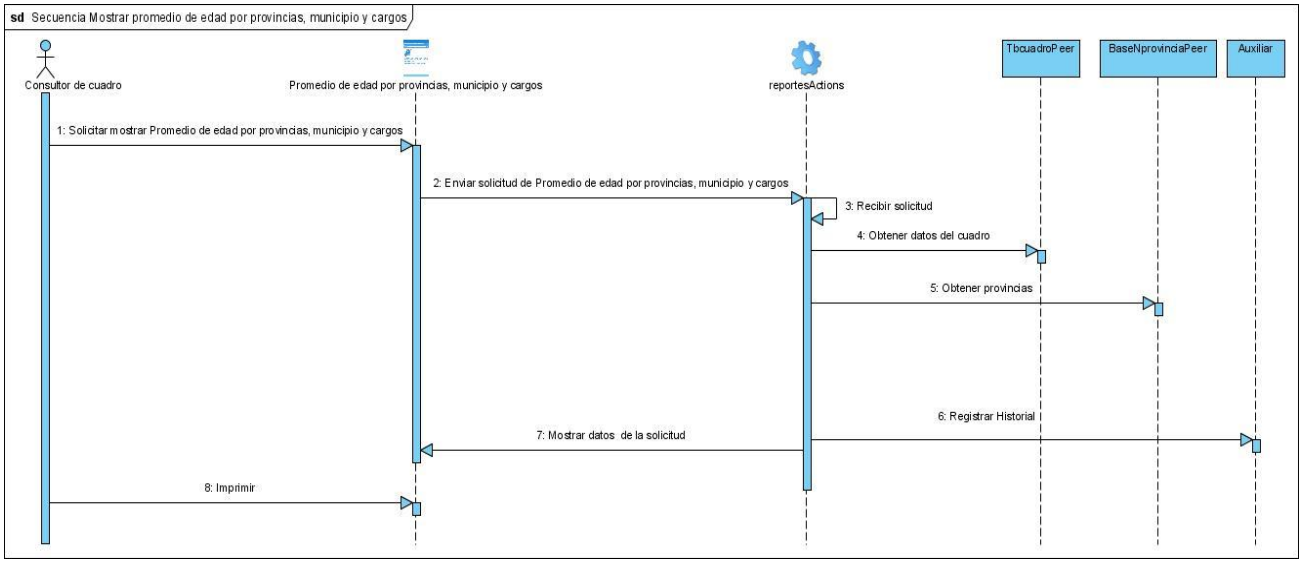

Figura 15. Secuencia: Mostrar Promedio de Edad por Provincias, Municipio y Cargos.

Anexo 7. Diagramas de Clases del Diseño del Módulo Gestión de Cuadro.

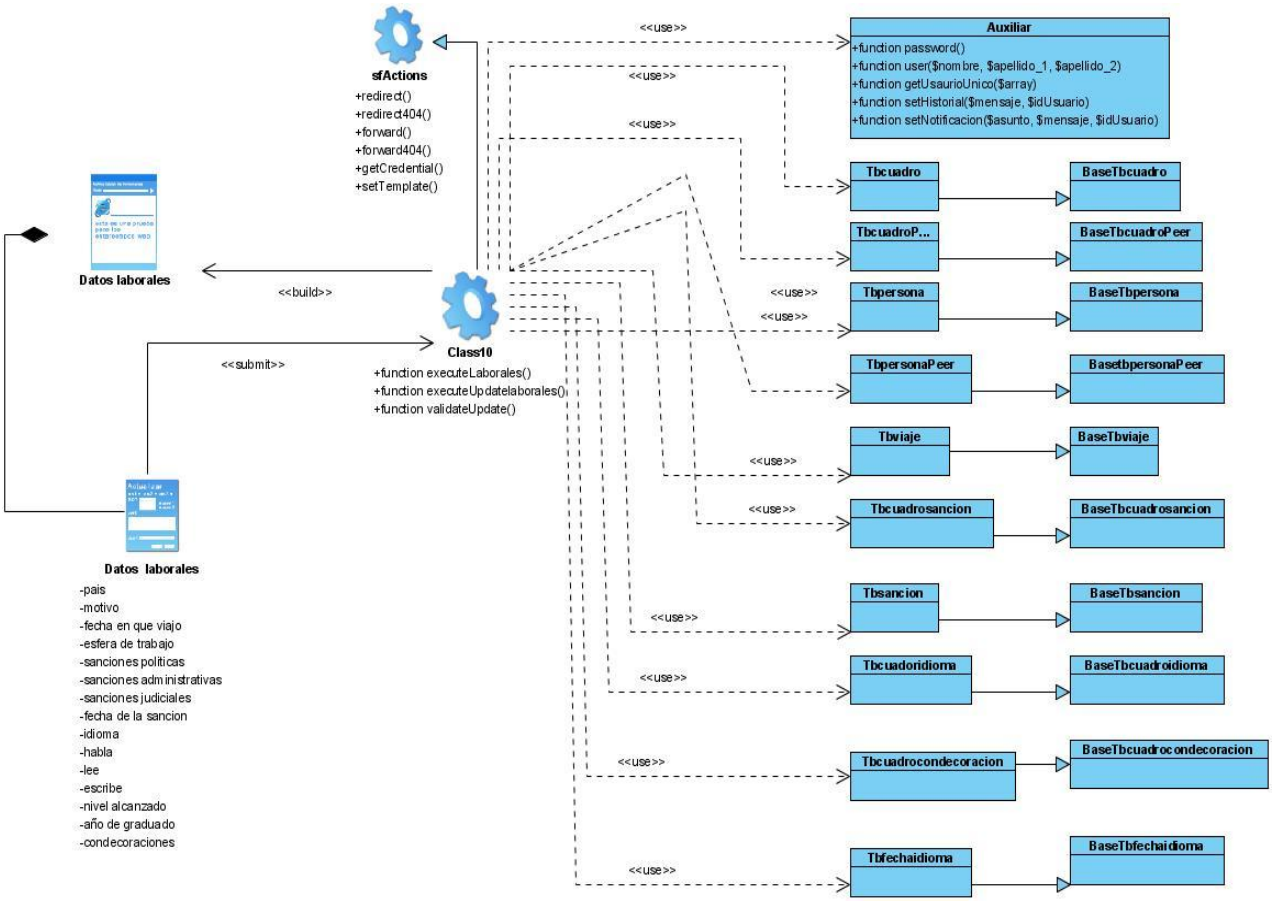

Figura 1. Diagramas de Clases: Registrar Datos Laborales del Cuadro.

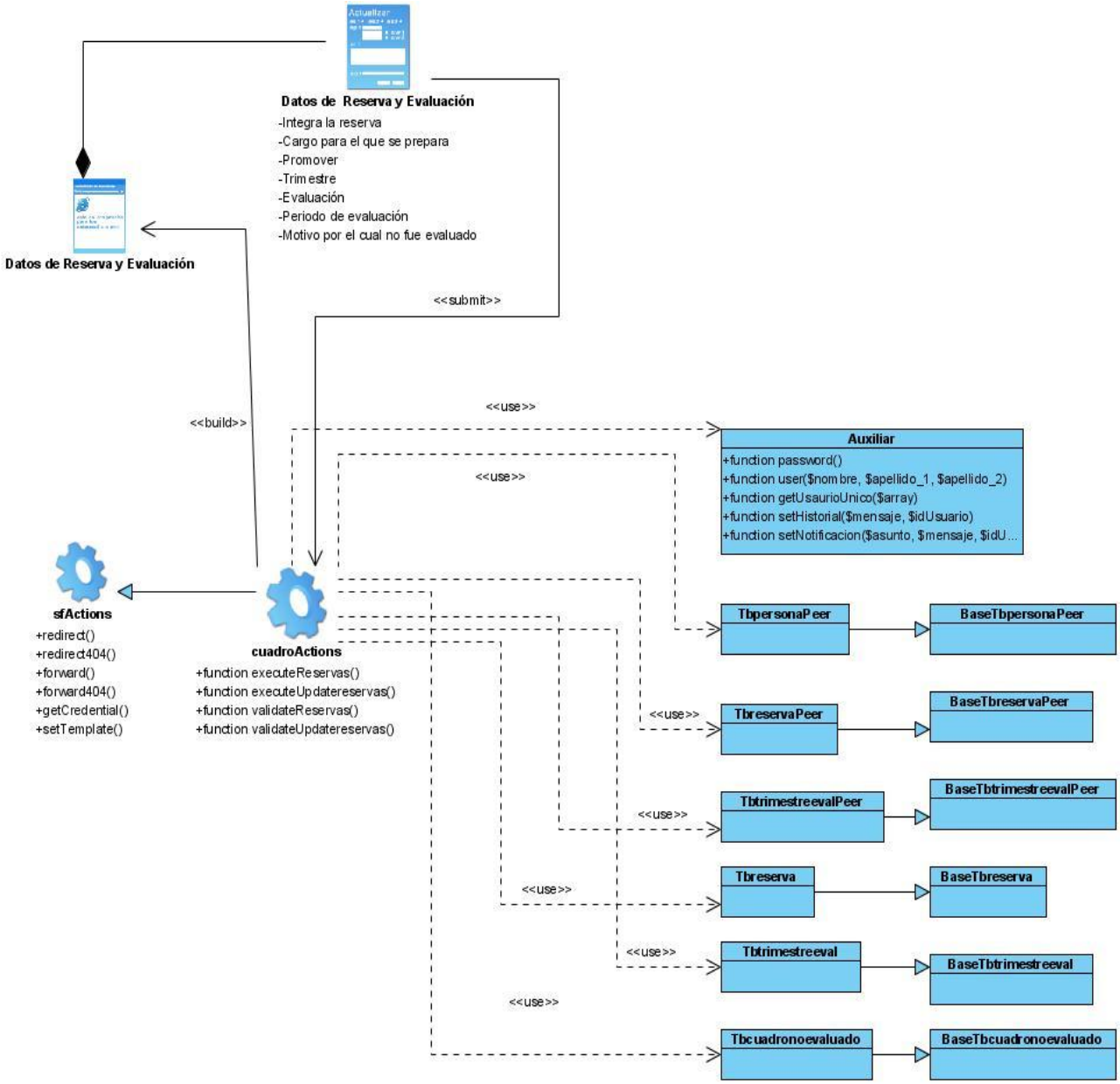

Figura 2. Diagramas de Clases: Registrar Datos de la Evaluación del Cuadro.

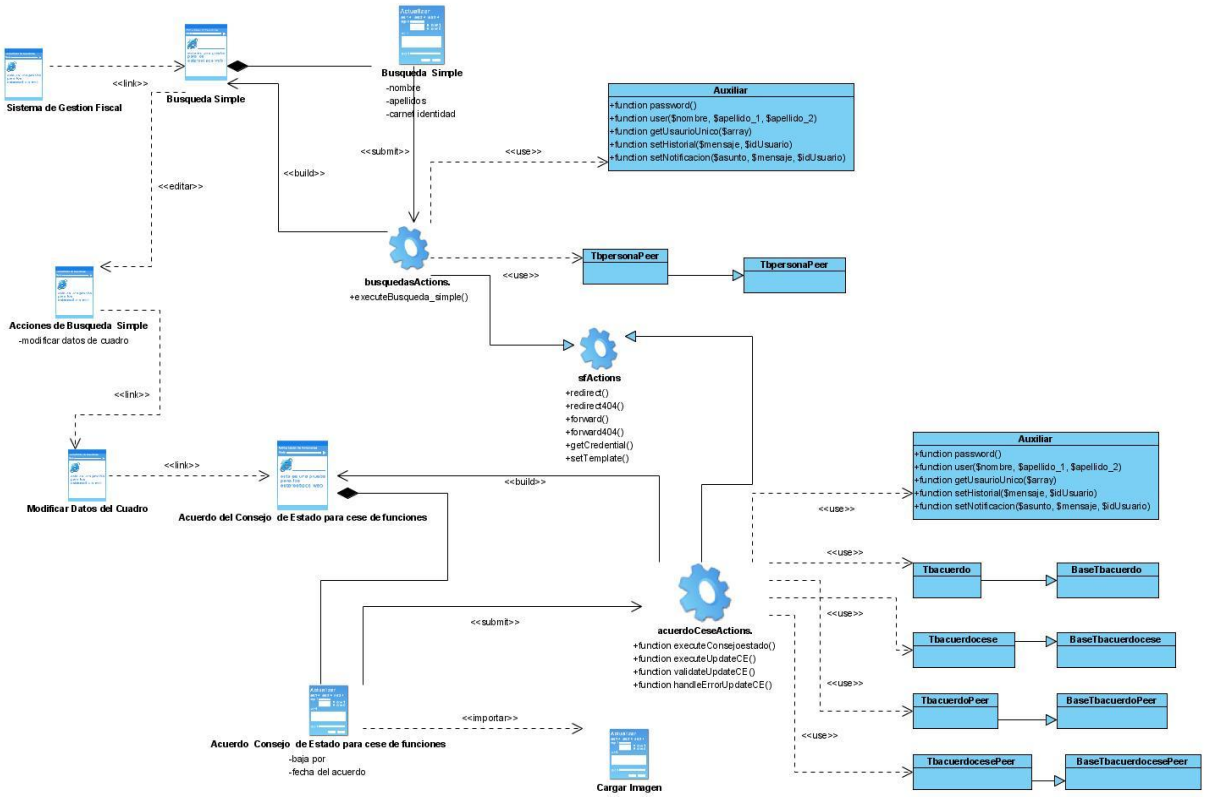

Figura 3. Diagramas de Clases: Registrar Acuerdo del Consejo de Estado para Cese de Funciones.

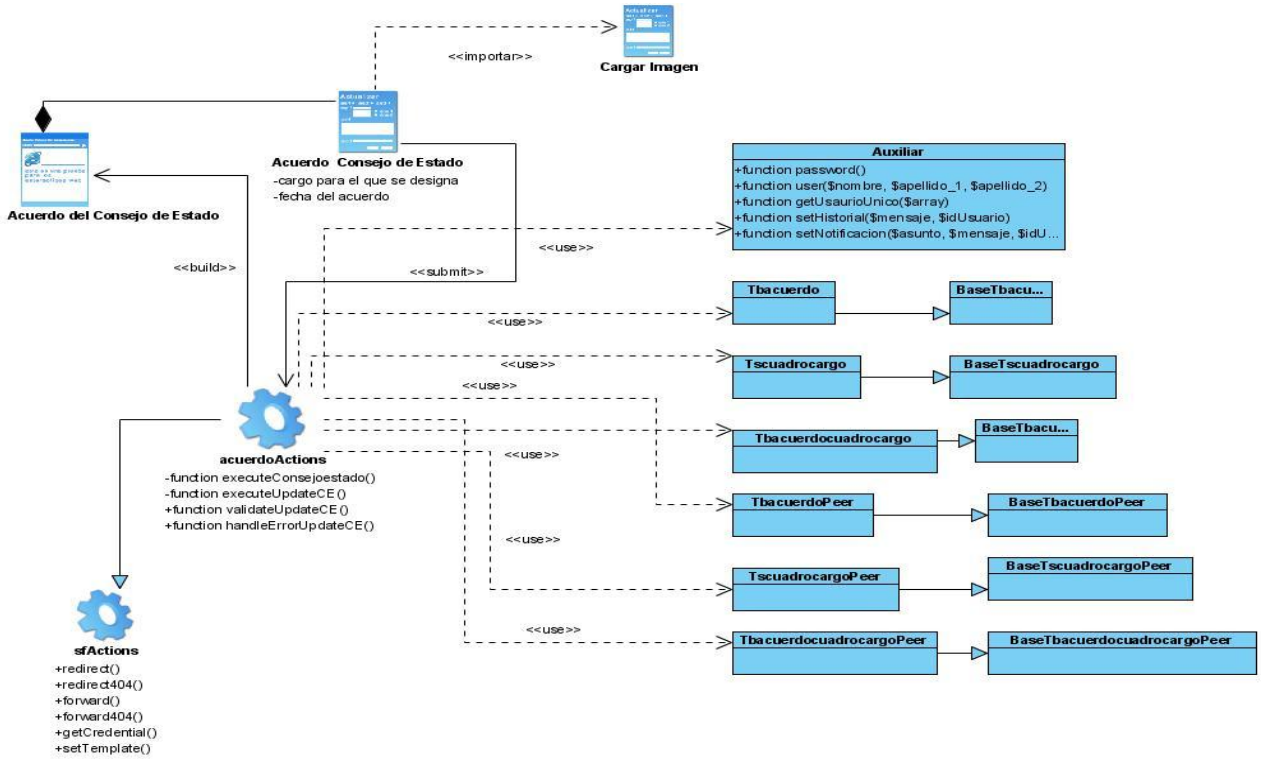

Figura 4. Diagramas de Clases: Registrar Acuerdo del Consejo de Estado.

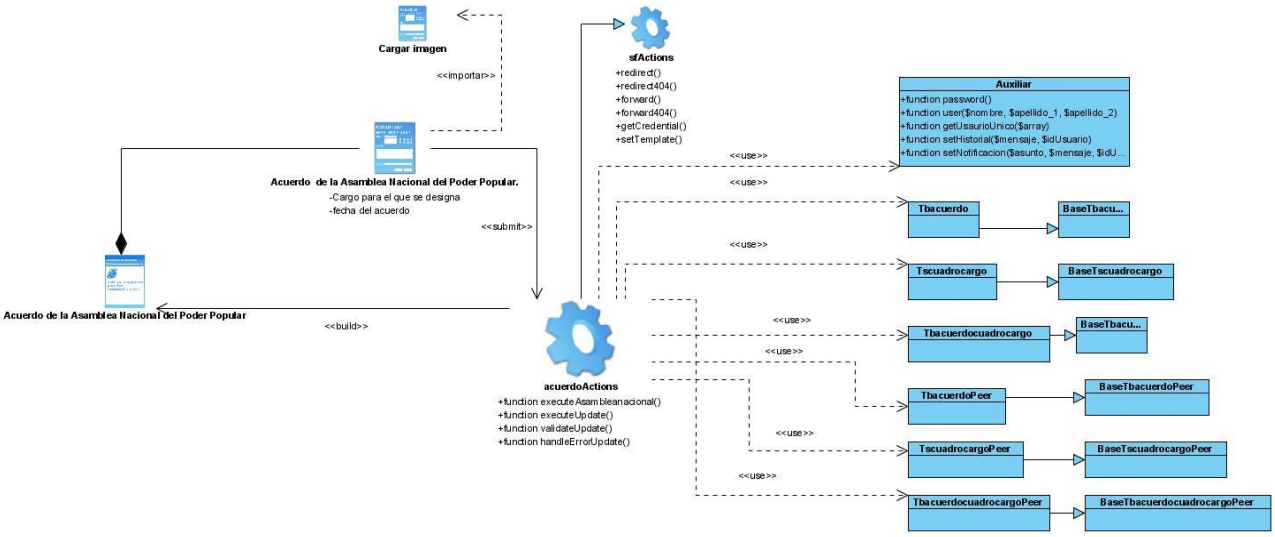

Figura 5. Diagramas de Clases: Registrar Acuerdo de la Asamblea Nacional del Poder Popular.

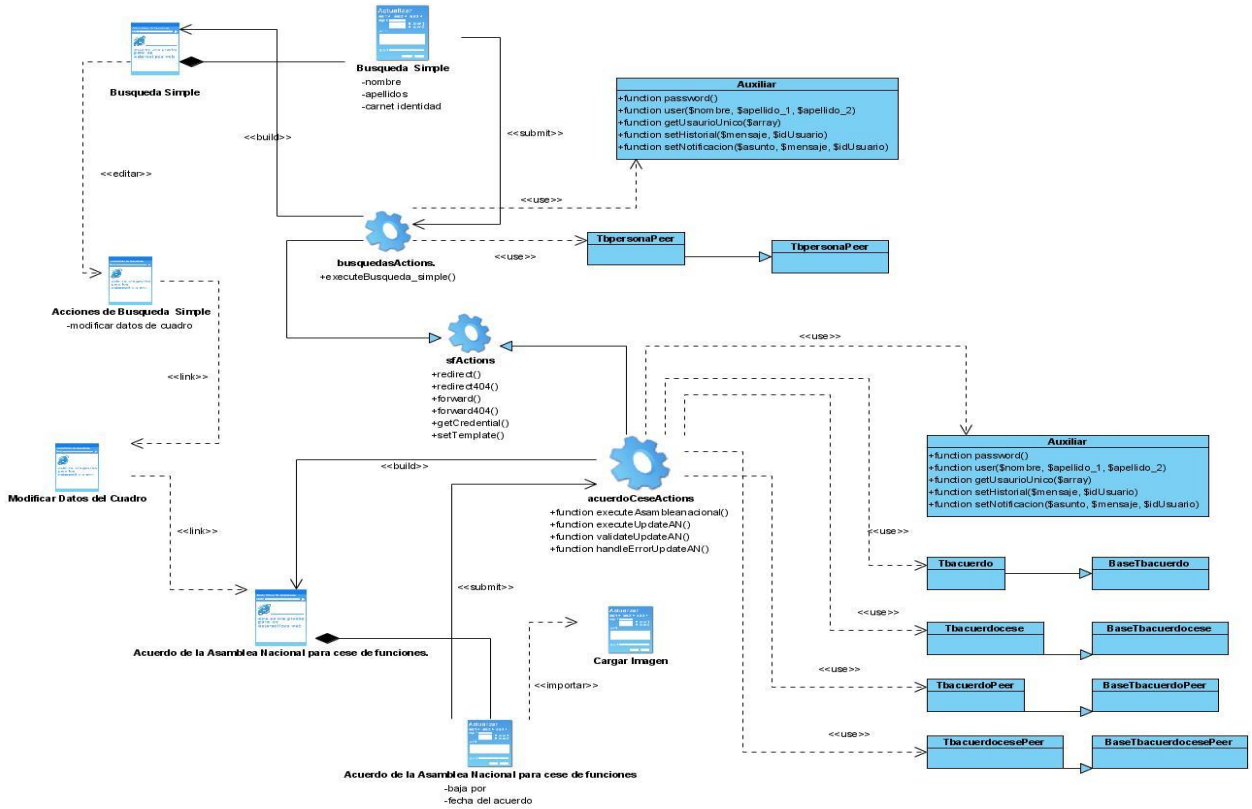

Figura 6. Diagramas de Clases: Registrar Acuerdo de la Asamblea Nacional del Poder Popular para Cese de Funciones.

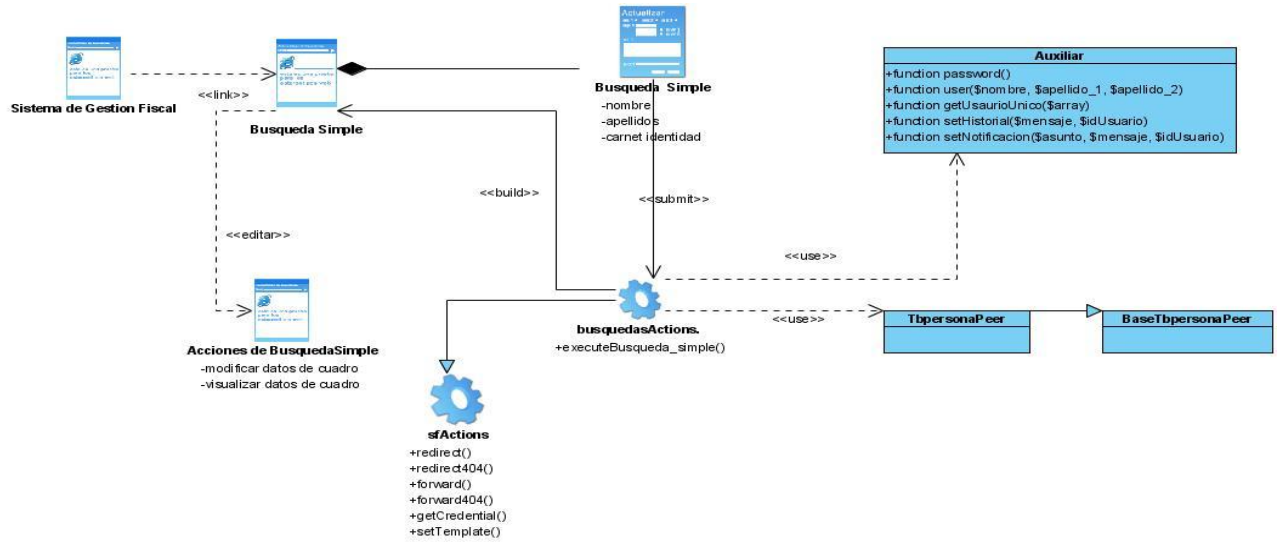

Figura 7. Diagramas de Clases: Búsqueda Simple.

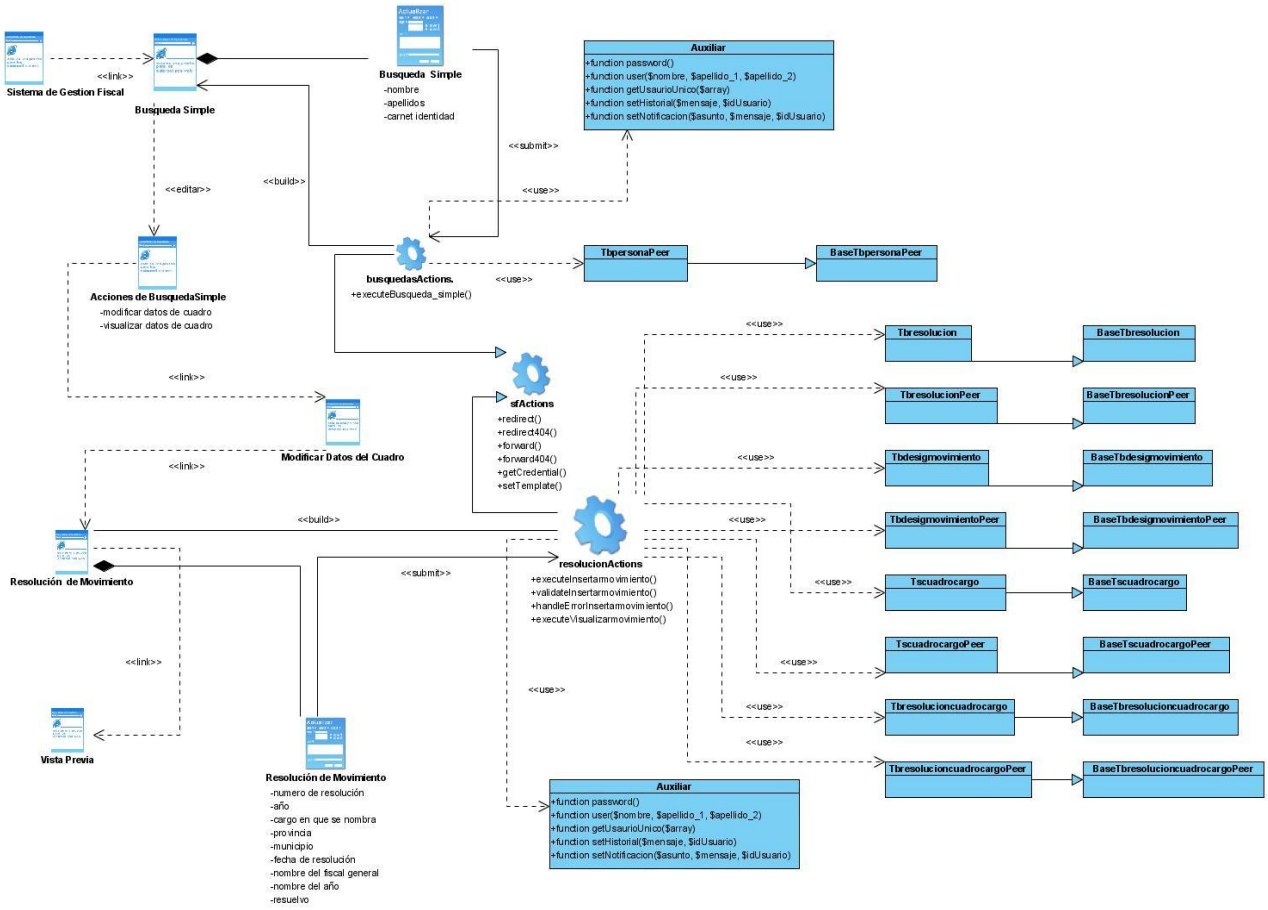

Figura 8. Diagramas de Clases: Registrar Resolución de Movimiento.

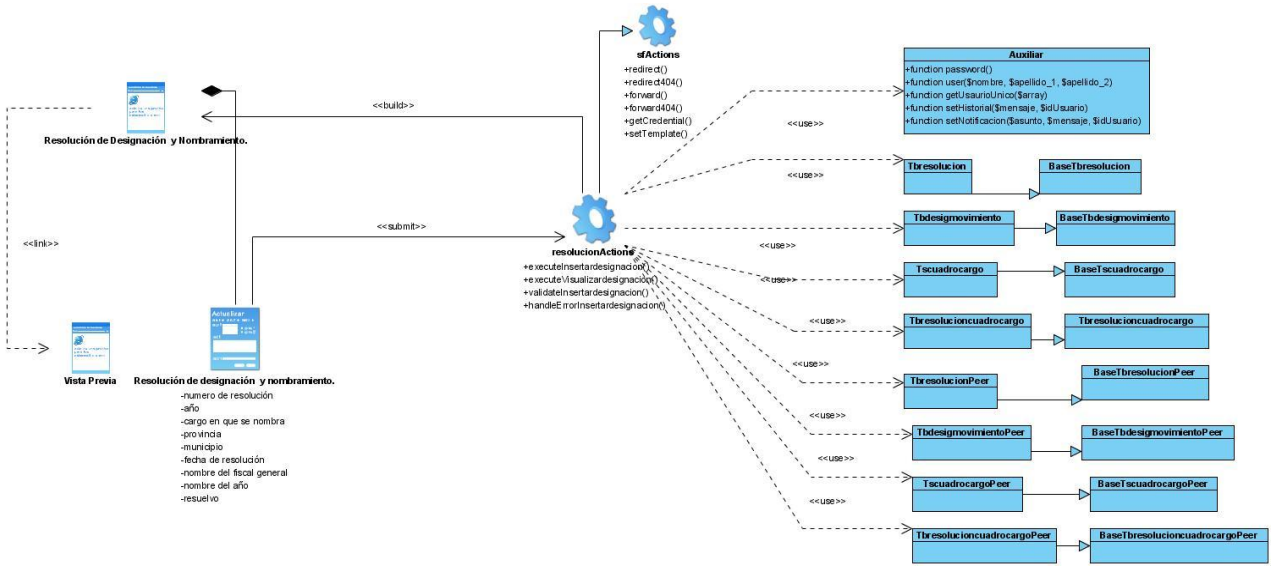

Figura 9. Diagramas de Clases: Registrar Resolución de Designación y Nombramiento.

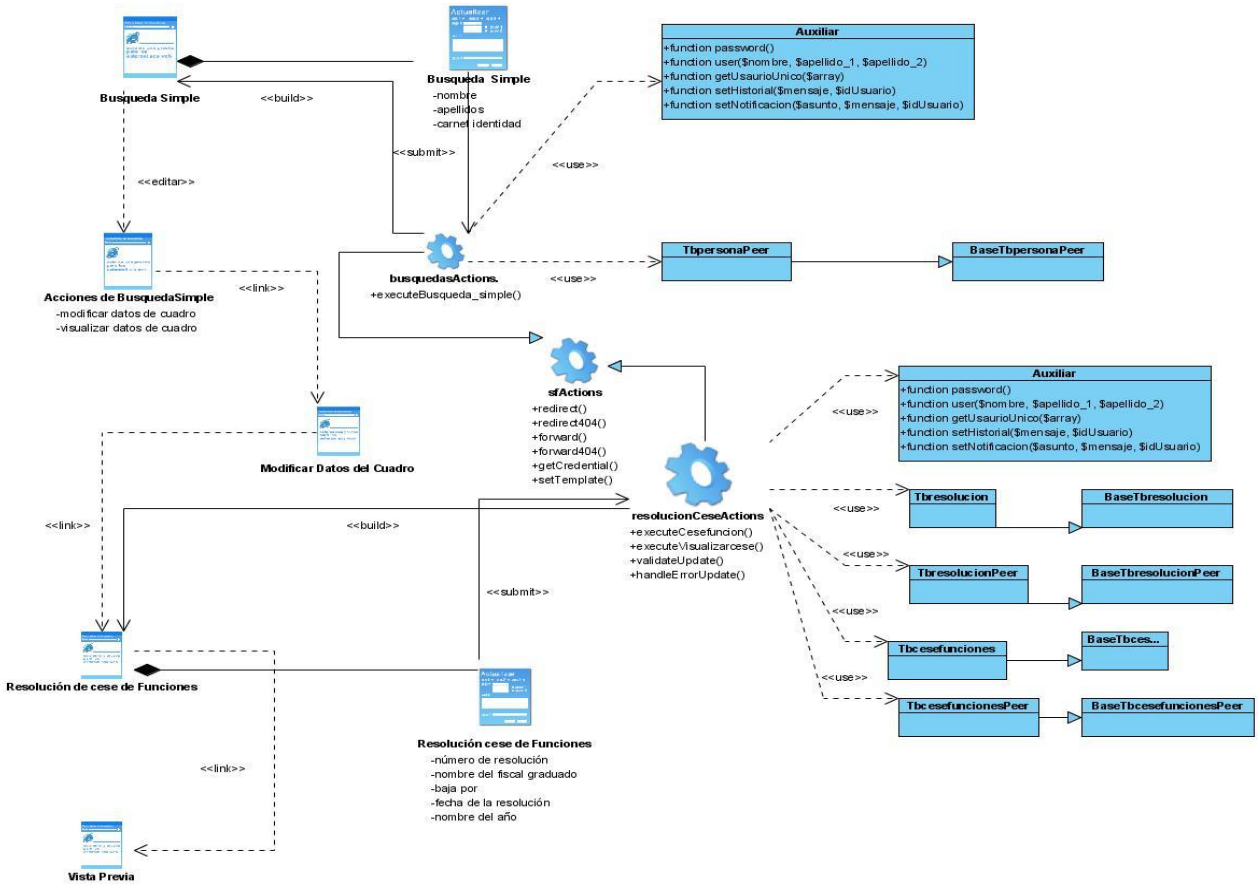

Figura 10. Diagramas de Clases: Registrar Resolución de Cese de las Funciones.

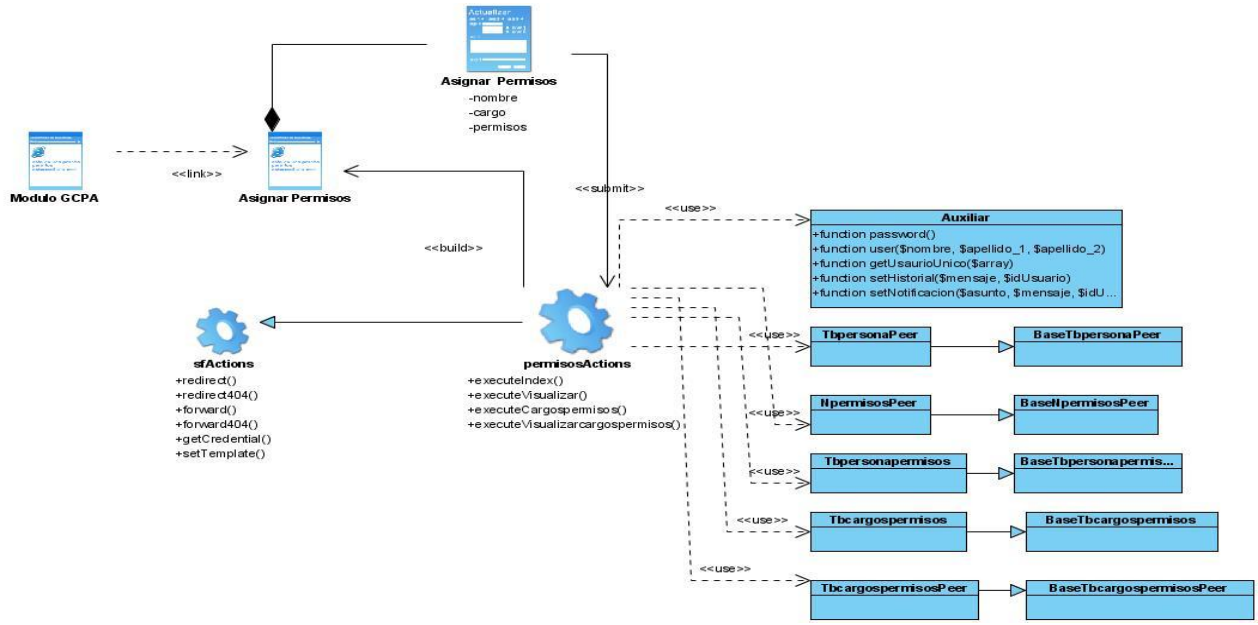

Figura 11. Diagramas de Clases: Registrar Permisos.

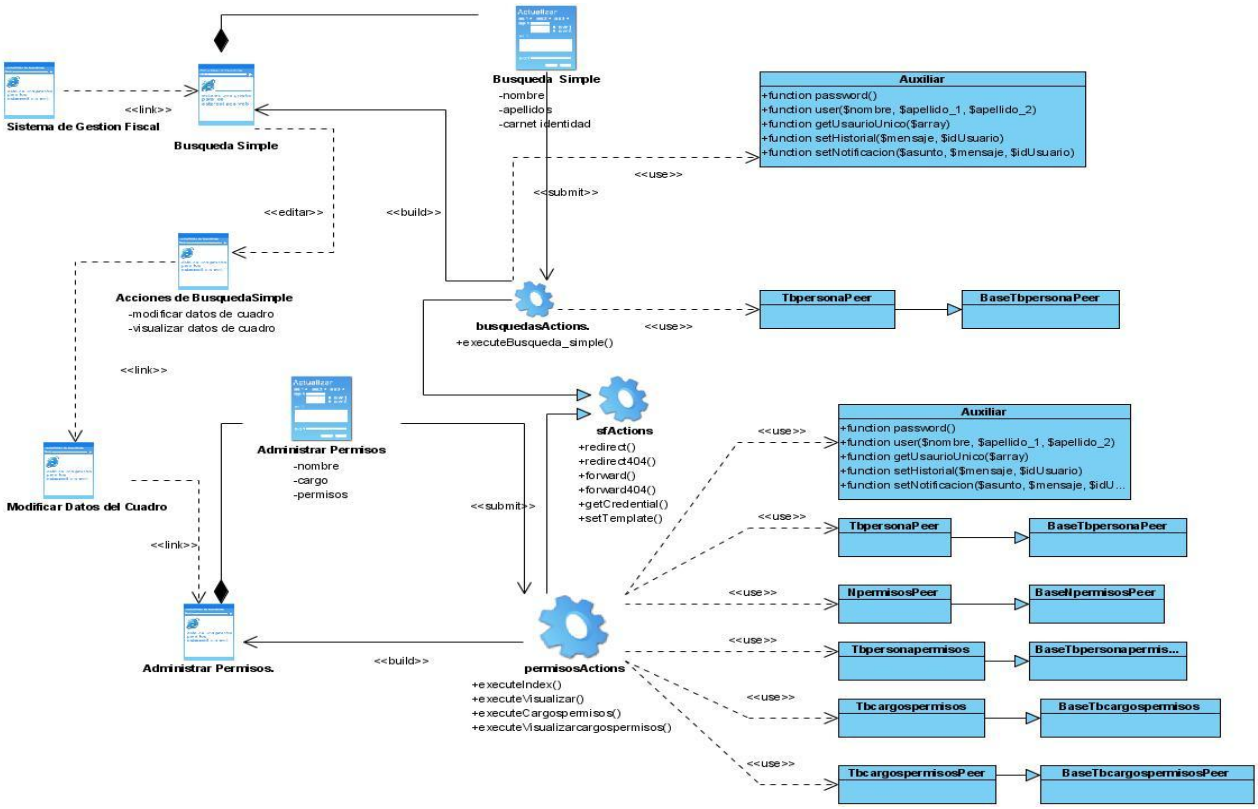

Figura 12. Diagramas de Clases: Modificar Permisos.

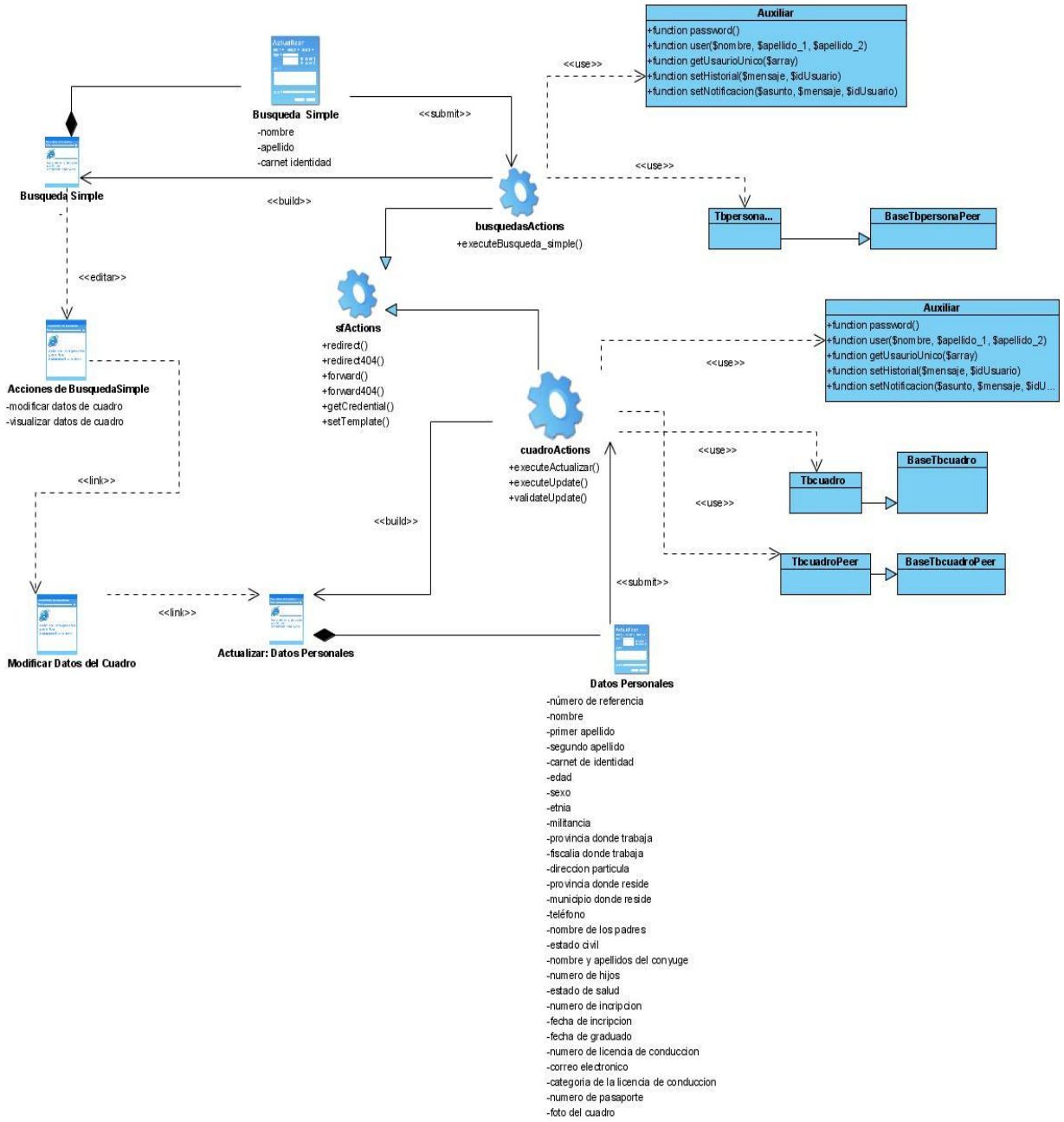

Figura 13. Diagramas de Clases: Modificar Datos Personales del Cuadro.

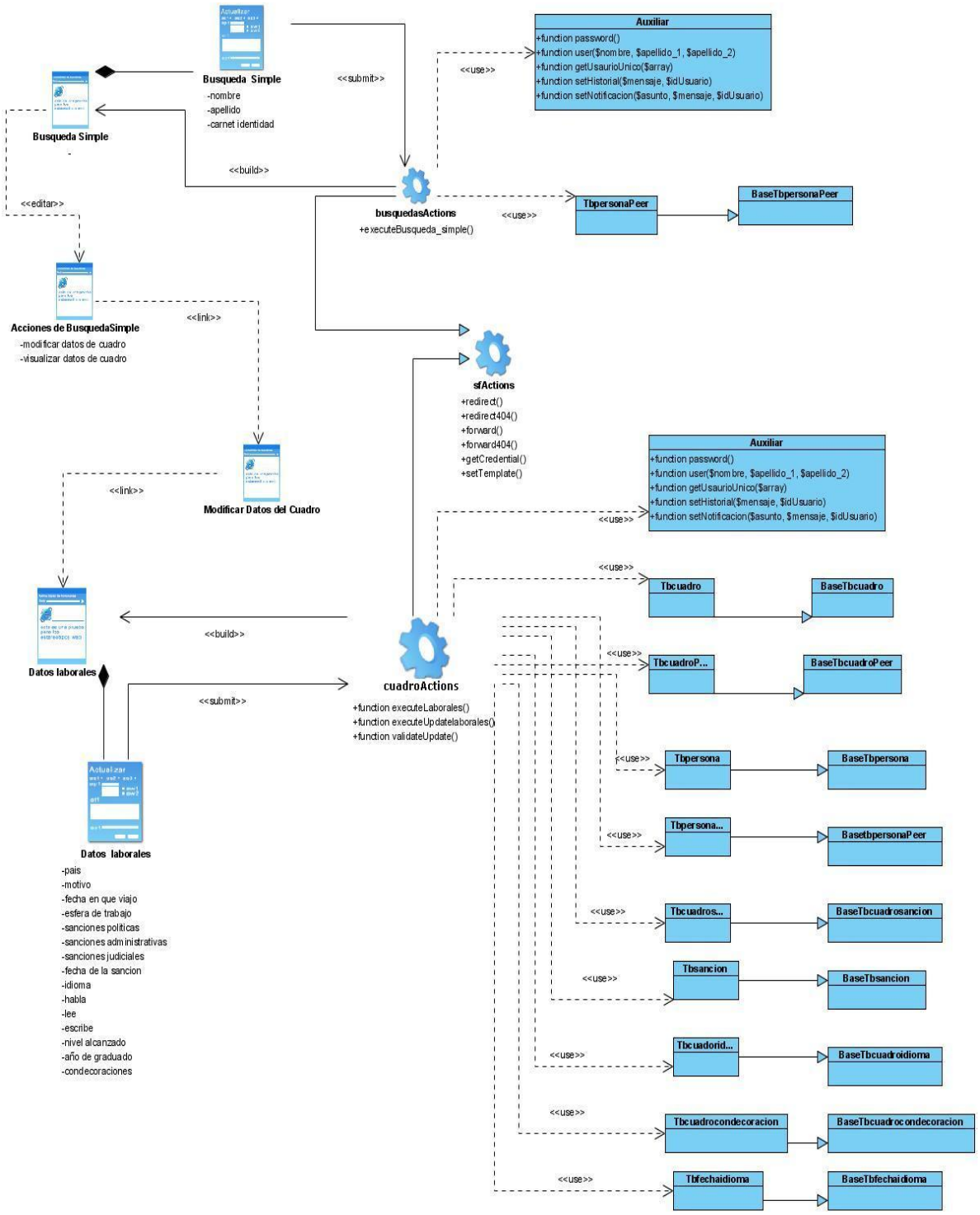

Figura 14. Diagramas de Clases: Modificar Datos Laborales del Cuadro.

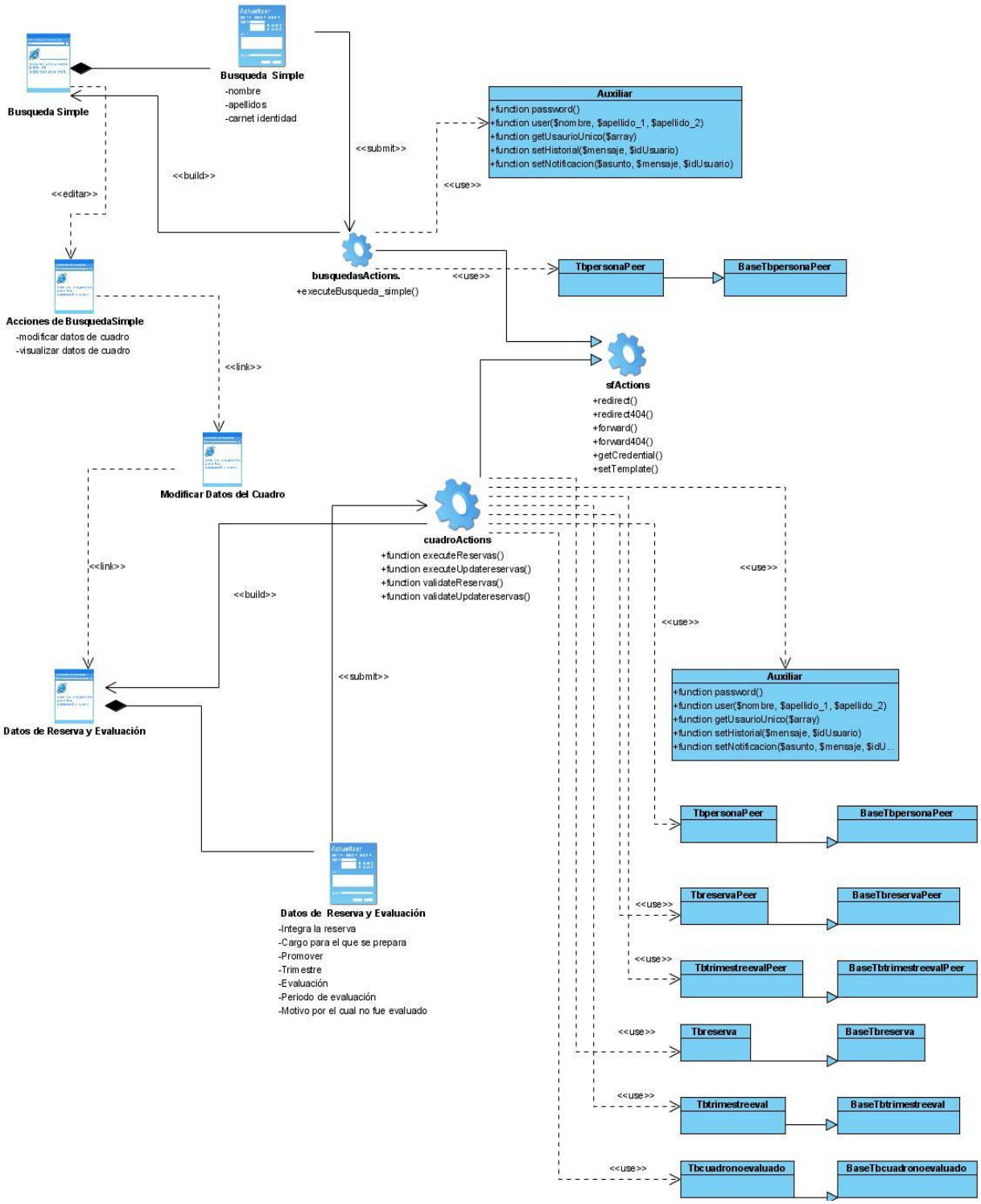

Figura 15. Diagramas de Clases: Modificar Datos de la Evaluación del Cuadro.

Anexo 8. Diagramas de Clases del Diseño del Módulo Personal de Apoyo.

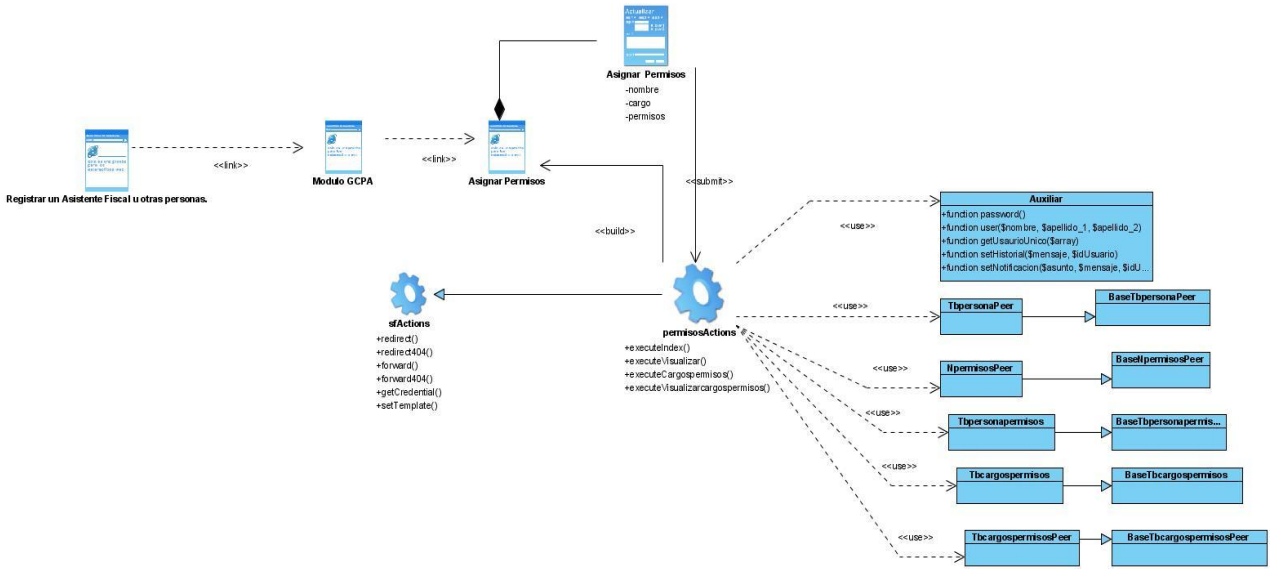

Figura 1. Diagramas de Clases: Registrar Permisos del Personal de Apoyo.

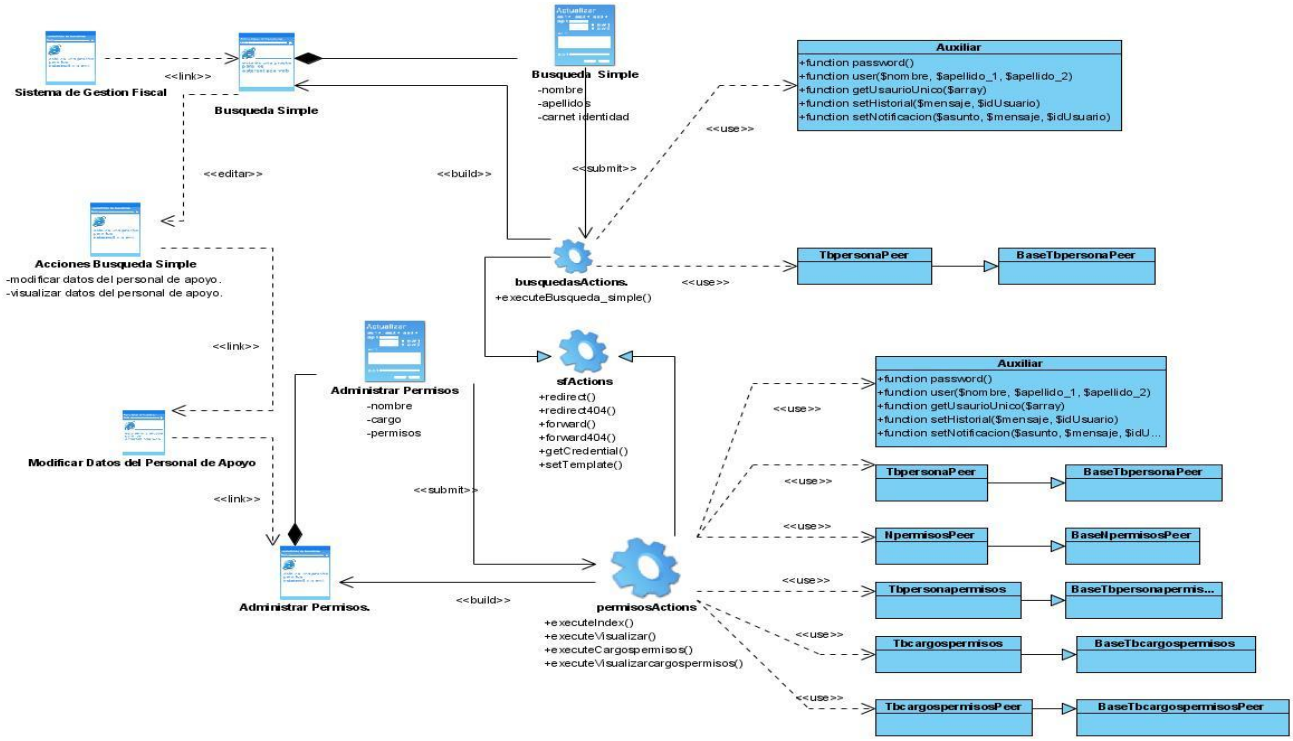

Figura 2. Diagramas de Clases: Modificar Permisos del Personal de Apoyo.

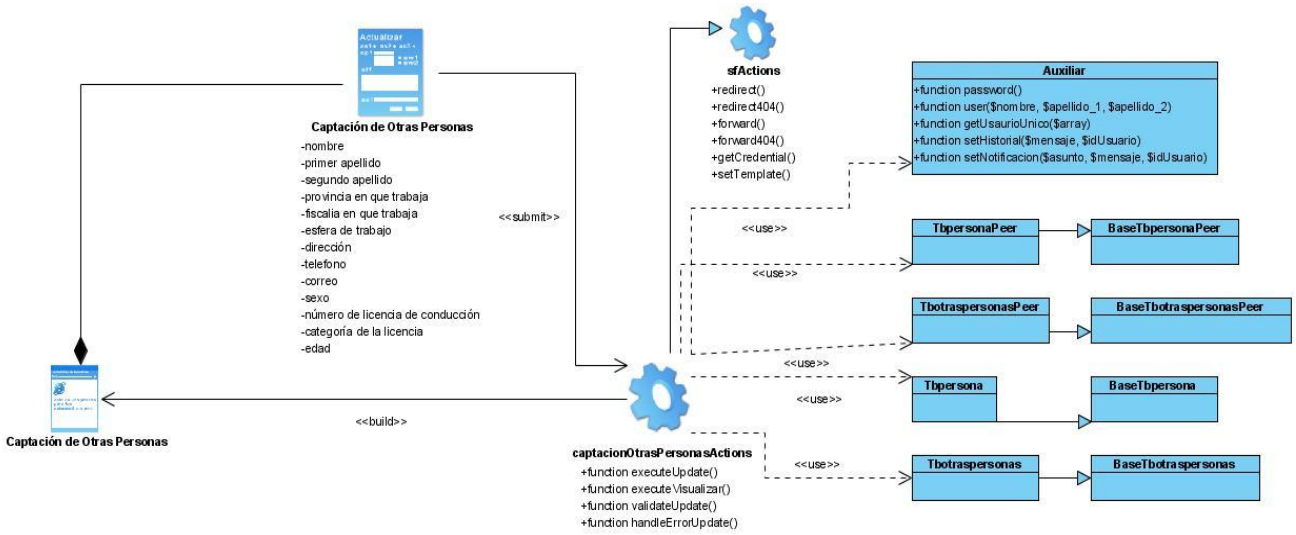

Figura 3. Diagramas de Clases: Registrar Datos de Captación de Otras Personas.

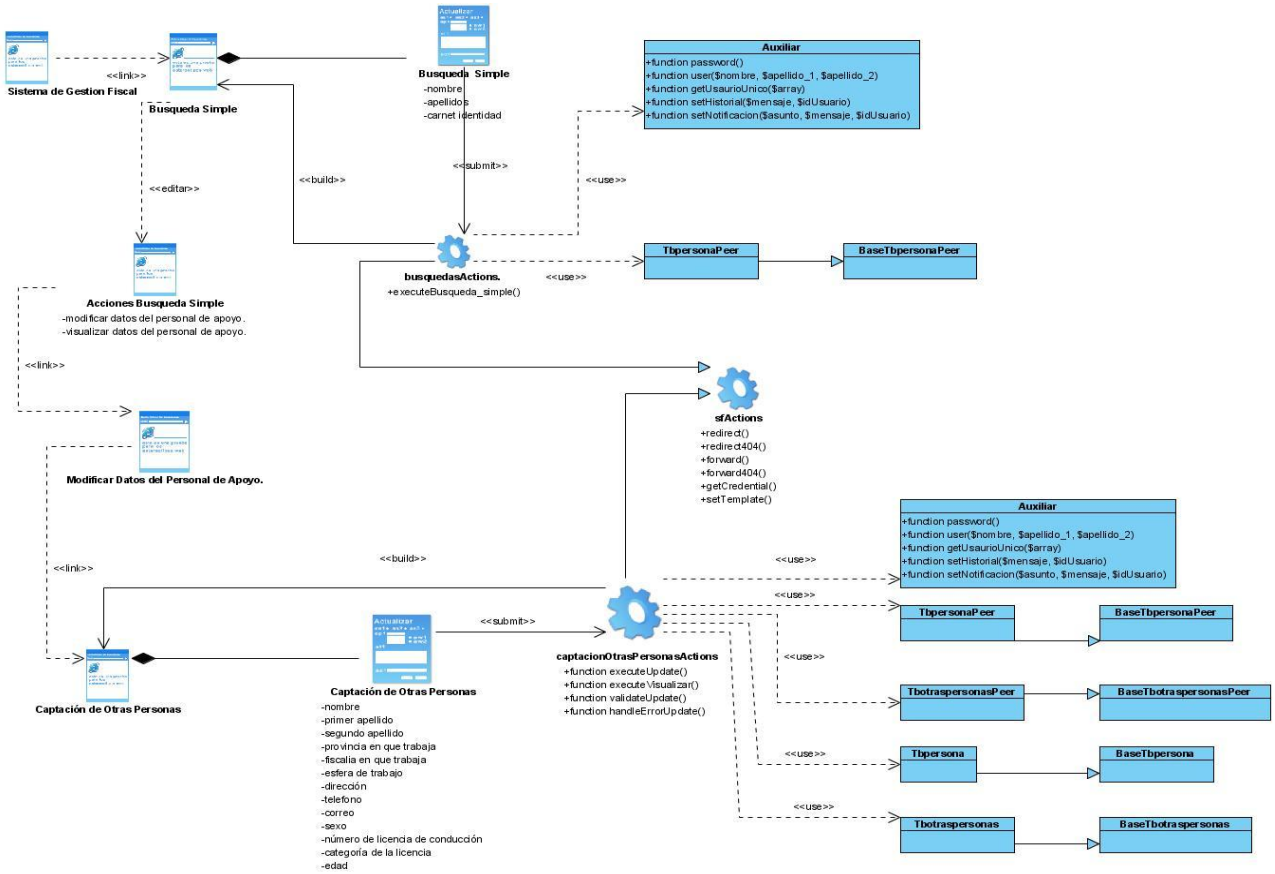

Figura 4. Diagramas de Clases: Modificar Datos de Captación de Otras Personas.

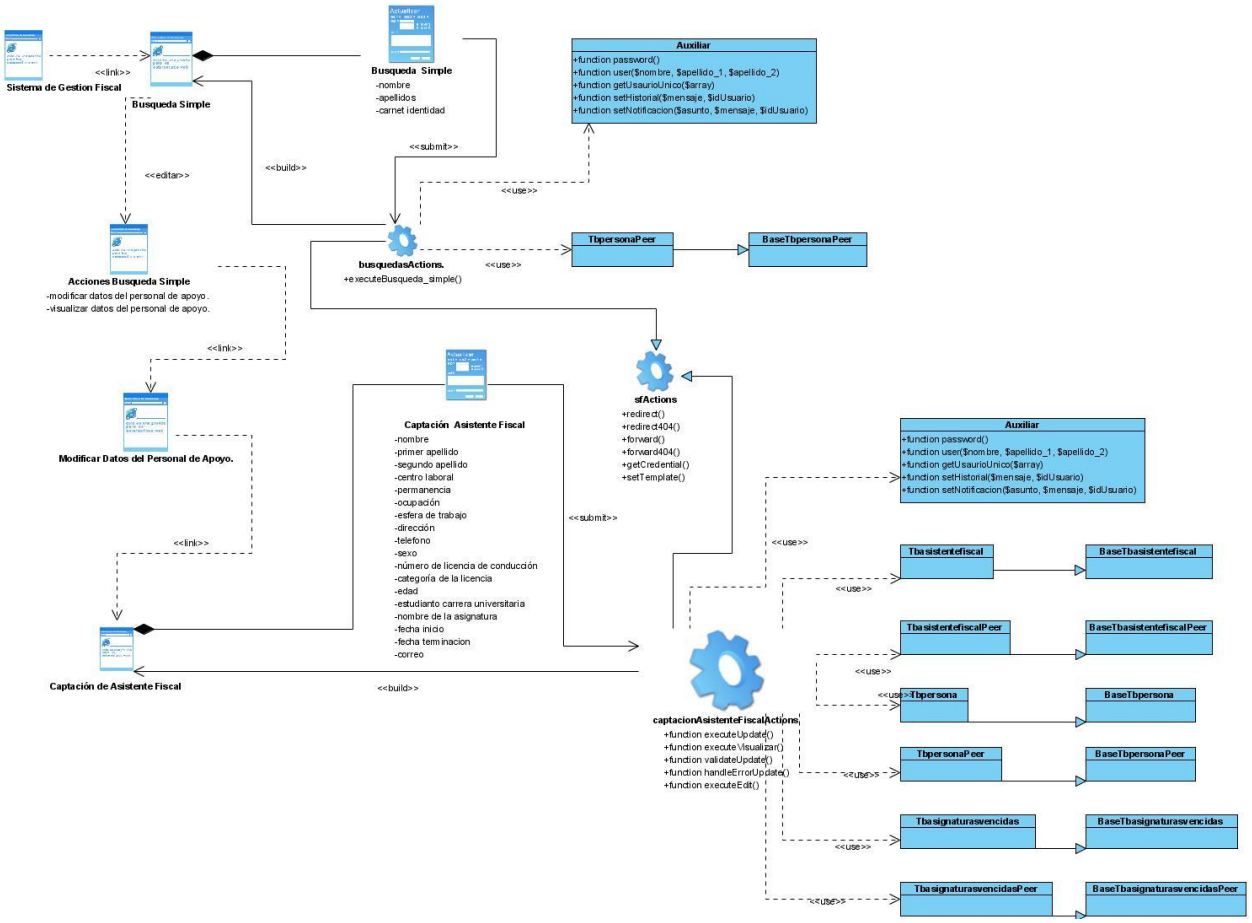

Figura 5. Diagramas de Clases: Modificar Datos de Captación del Asistente Fiscal.

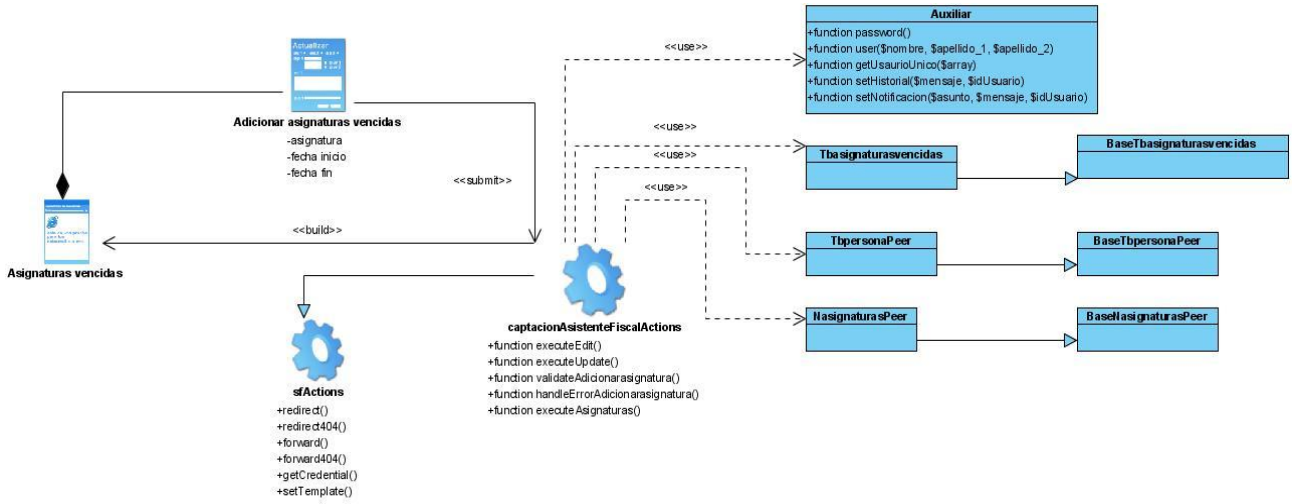

Figura 6. Diagramas de Clases: Adicionar Asignaturas Vencidas.

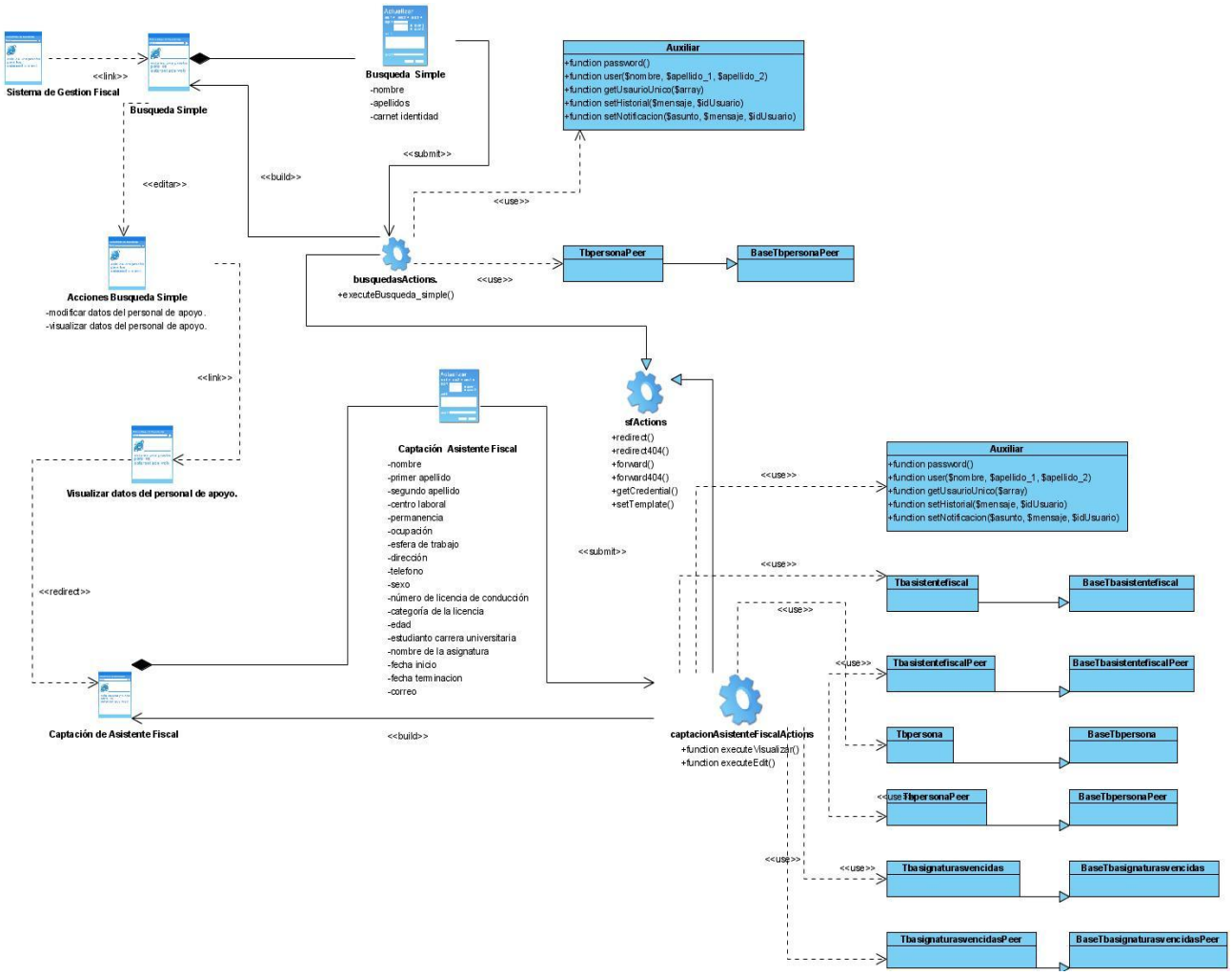

Figura 7. Diagramas de Clases: Visualizar Captación de Asistente Fiscal.

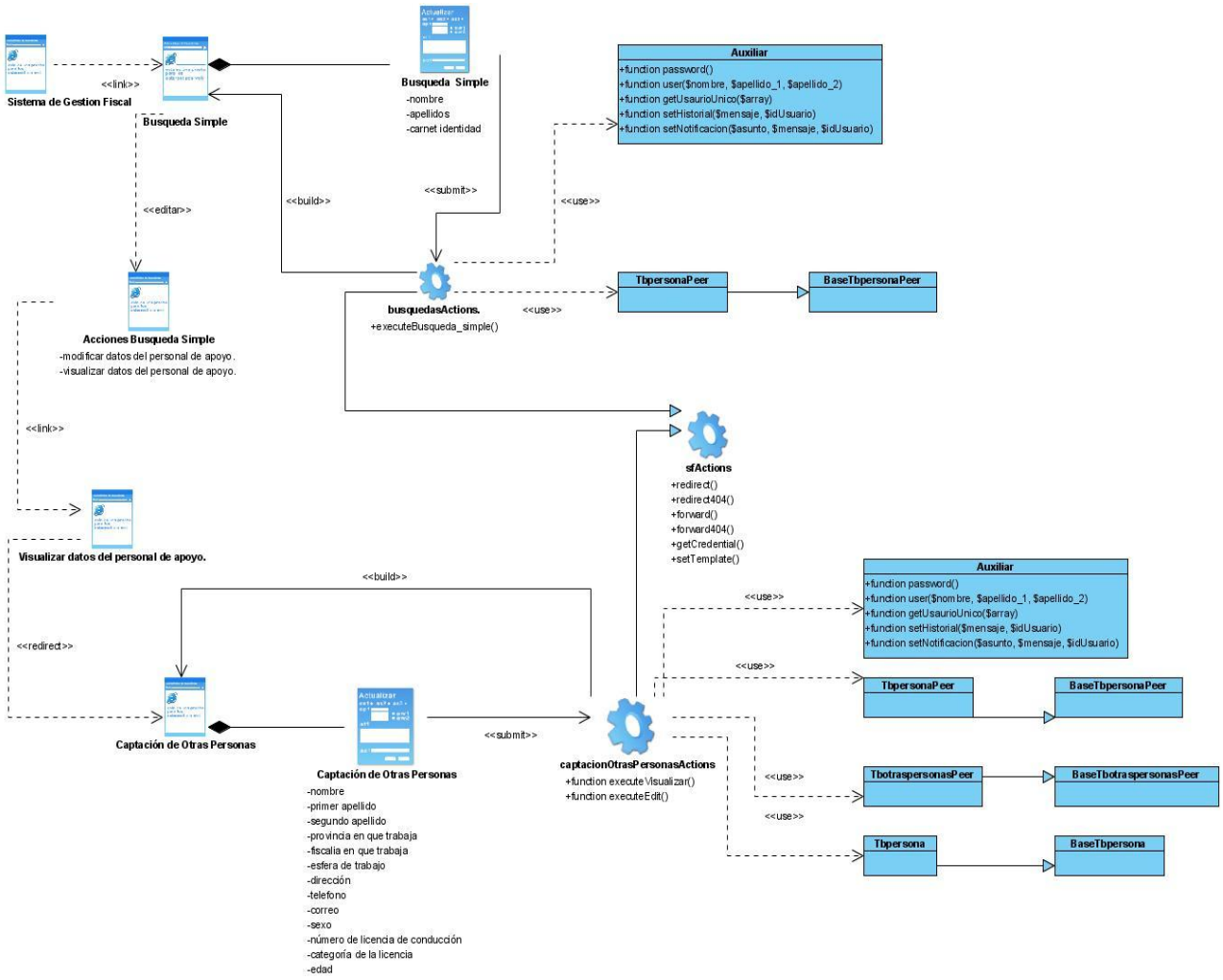

Figura 8. Diagramas de Clases: Visualizar Captación de Otras Personas.

Anexo 9. Diagramas de Clases del Diseño del Módulo Capacitación.

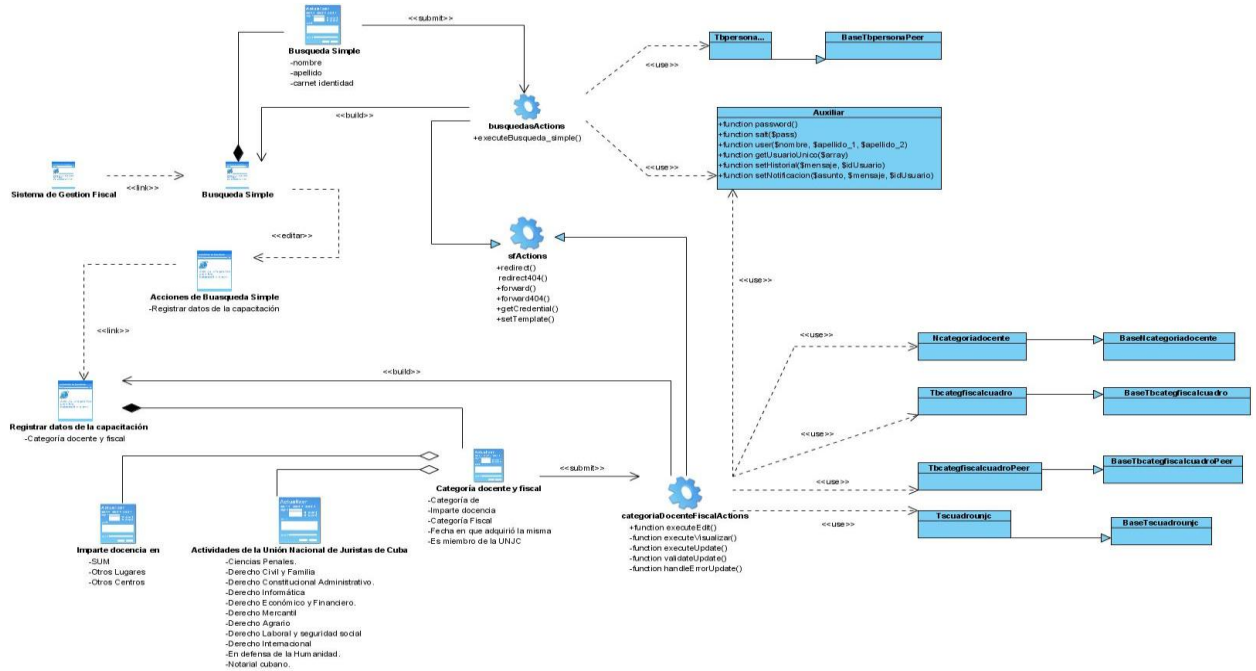

Figura 1. Diagramas de Clases: Registrar Categoría Docente y Fiscal del Cuadro.

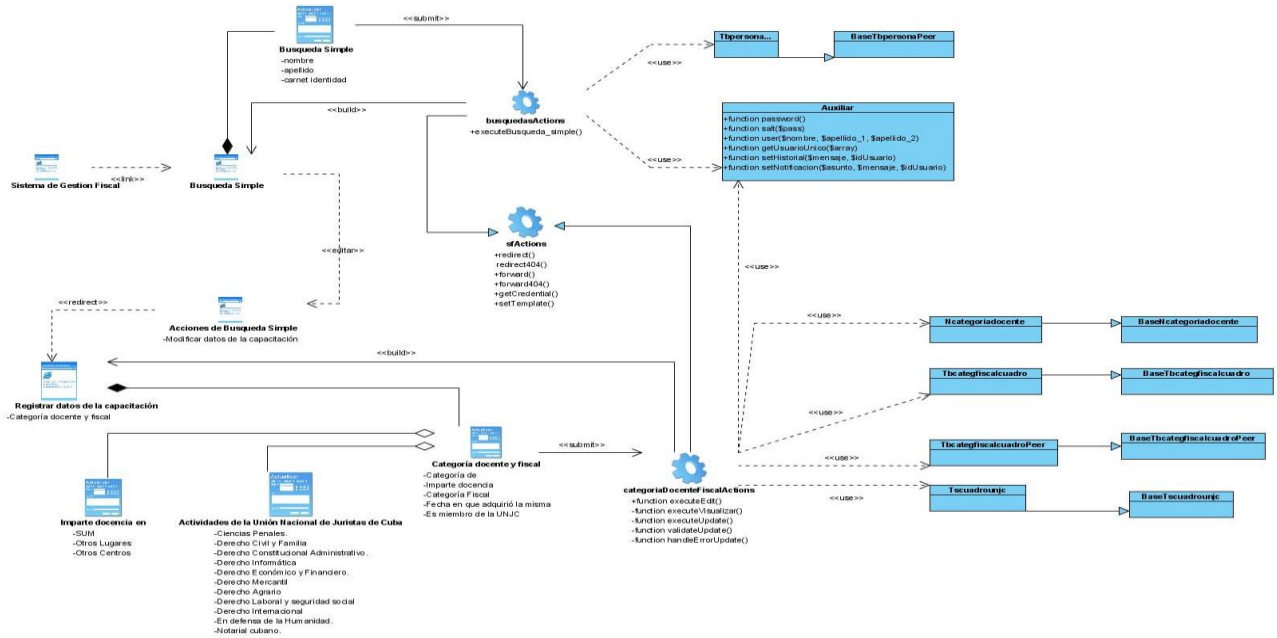

Figura 2. Diagramas de Clases: Modificar Categoría Docente y Fiscal.

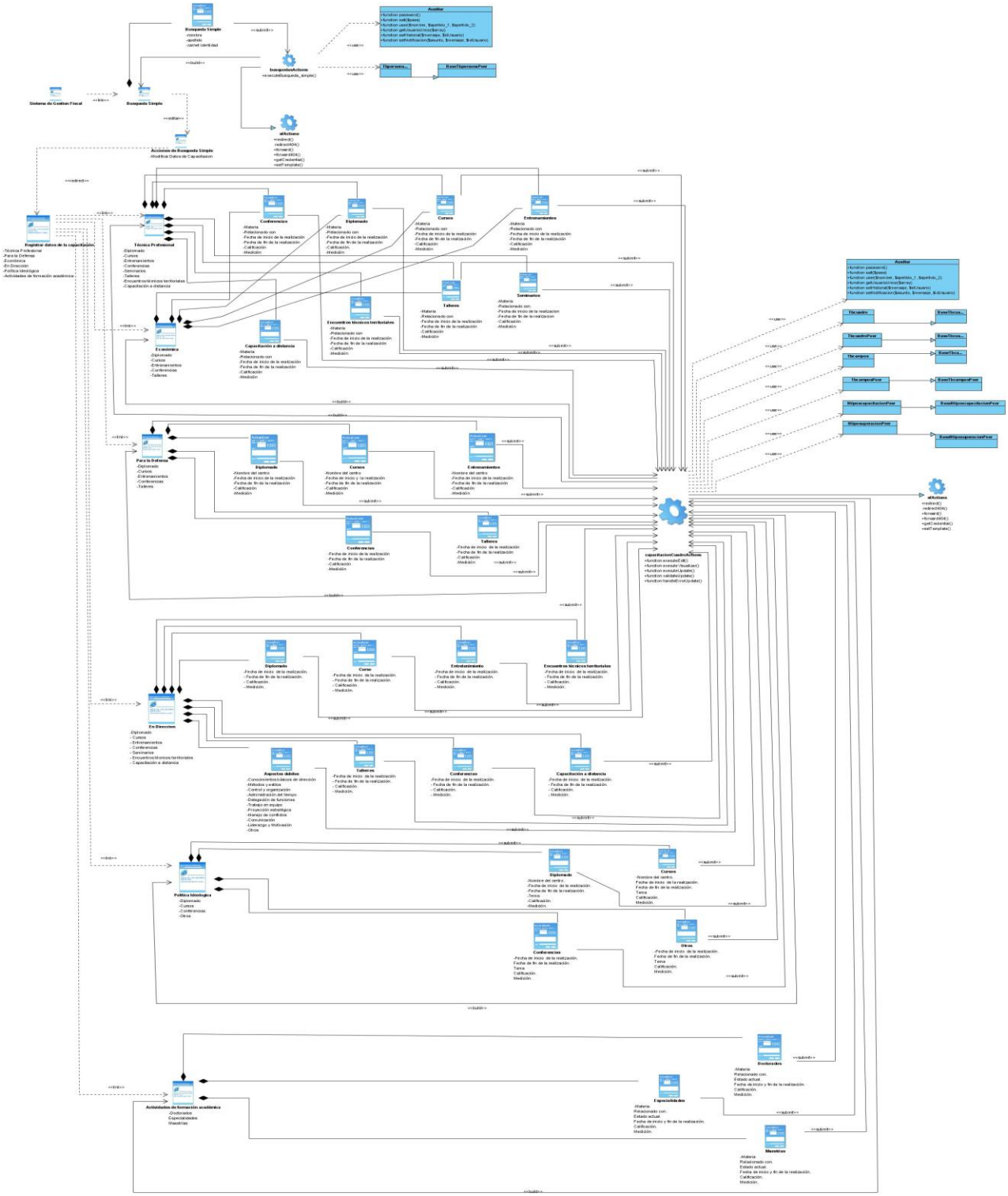

Figura 3. Diagramas de Clases: Modificar Datos de Capacitación.

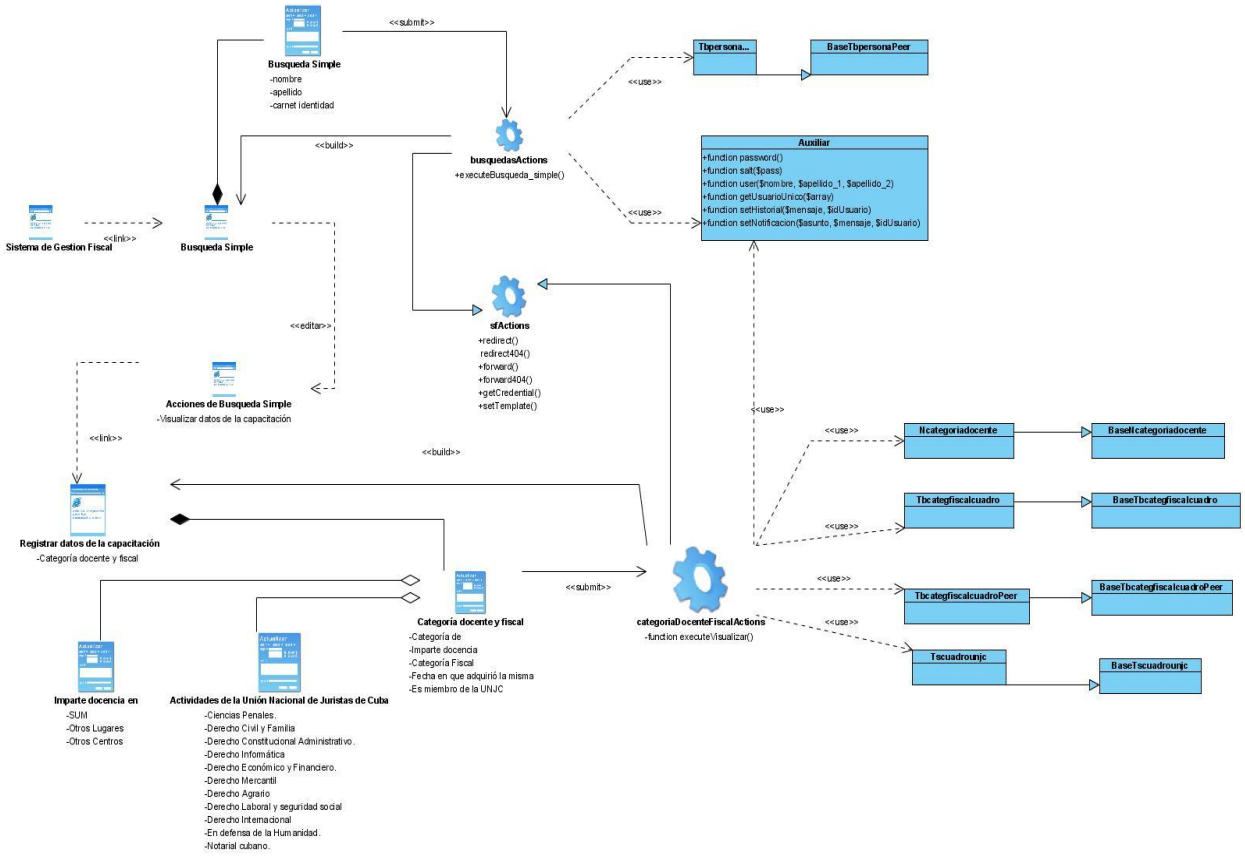

Figura 4. Diagramas de Clases: Visualizar Categoría Docente y Fiscal.

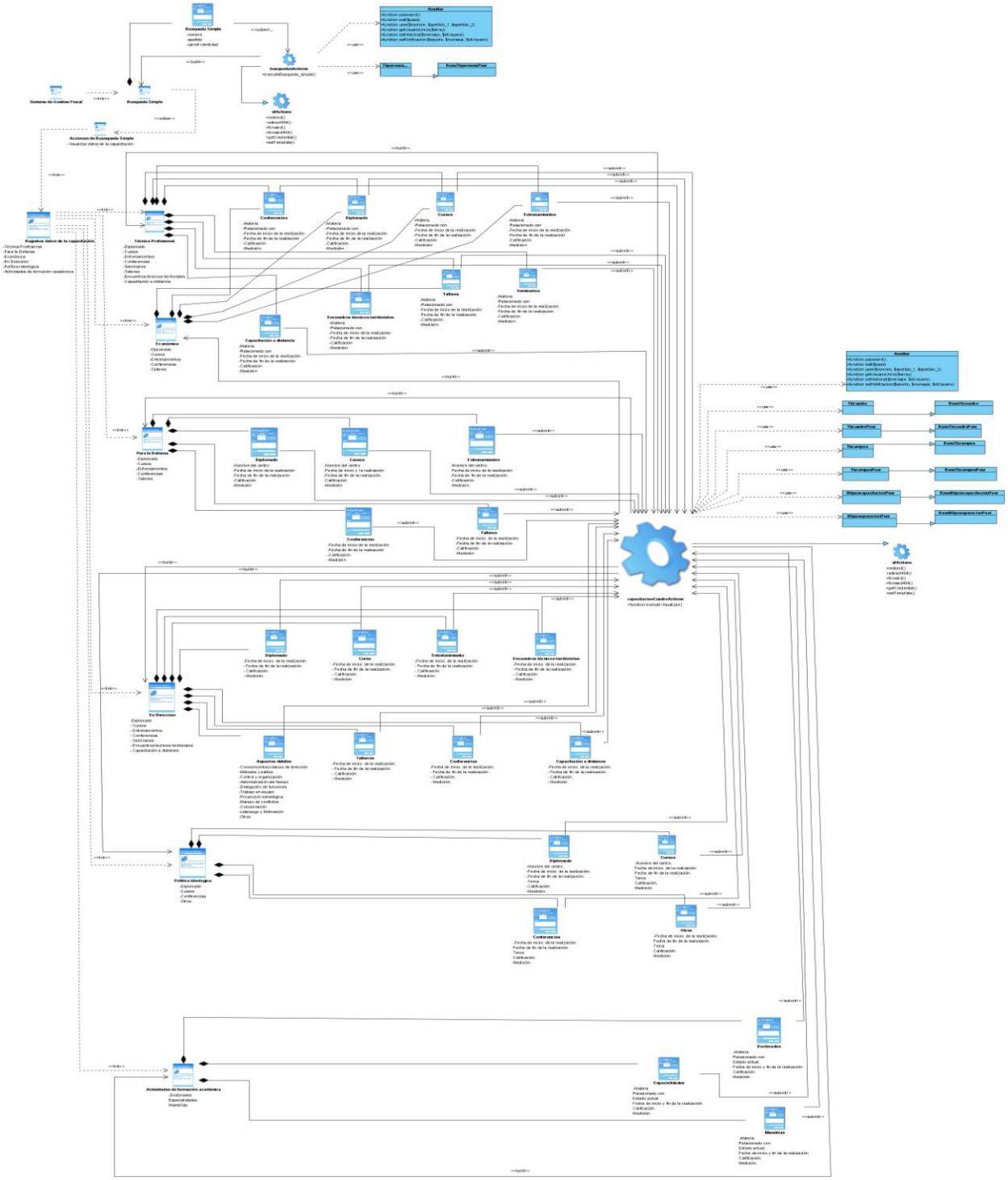

Figura 5. Diagramas de Clases: Visualizar Datos de Capacitación.

Anexo 10. Diagramas de Clases del Diseño del Módulo Captación de Plantilla.

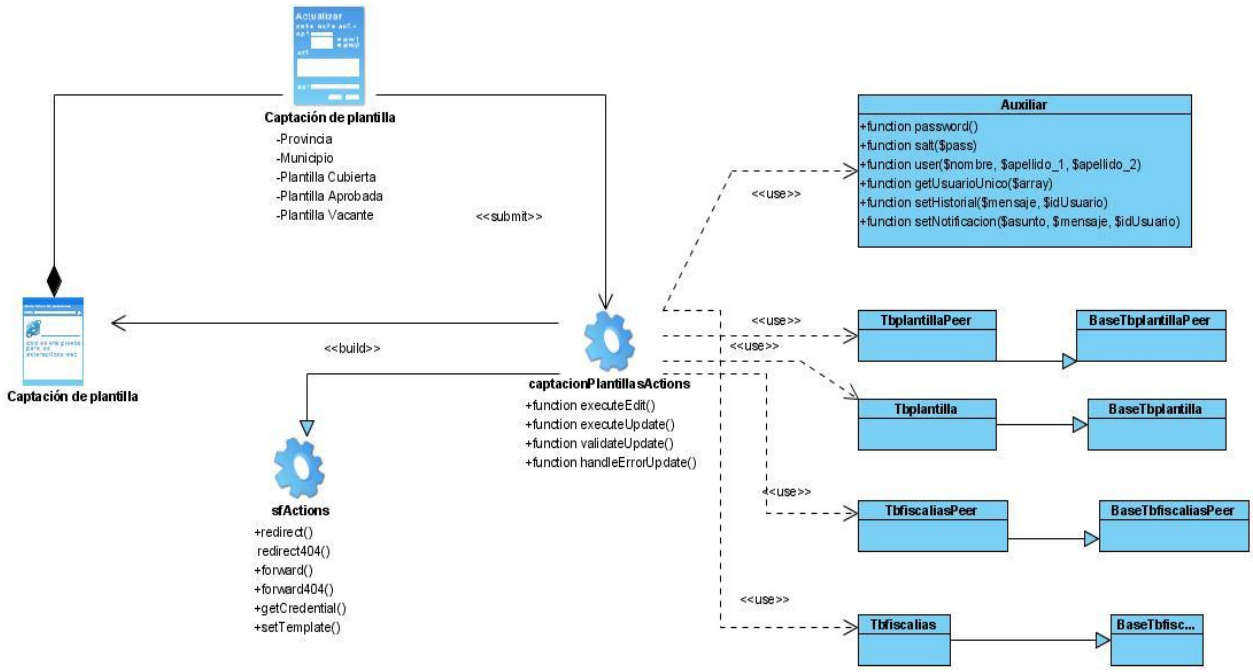

Figura 1. Diagramas de Clases: Modificar Datos de Captación de Plantilla.

Anexo 11. Diagramas de Clases del Diseño del Módulo Generalidades.

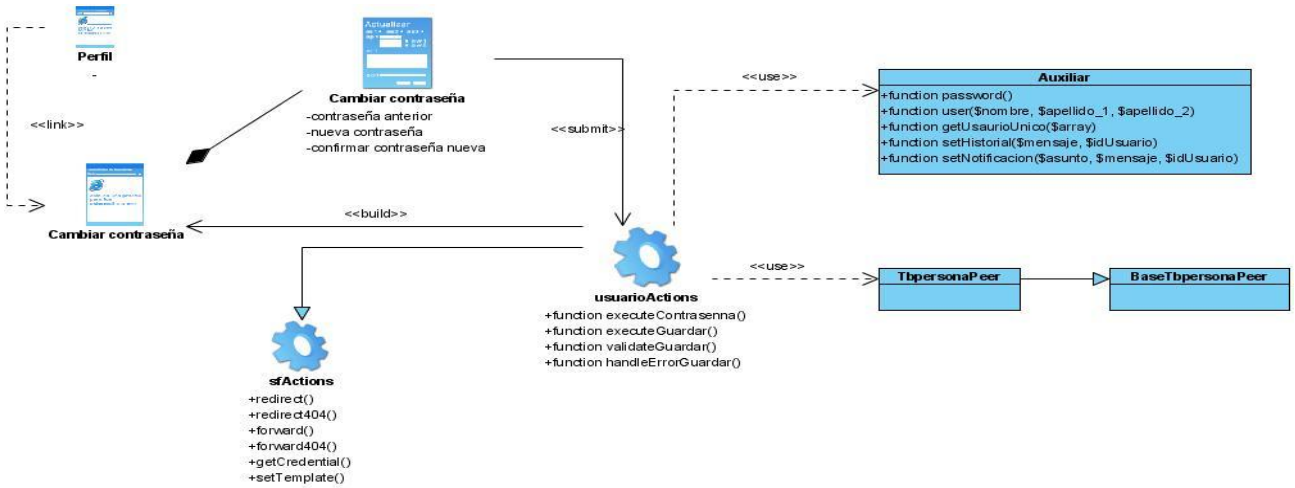

Figura 1. Diagramas de Clases: Cambiar Contraseña.

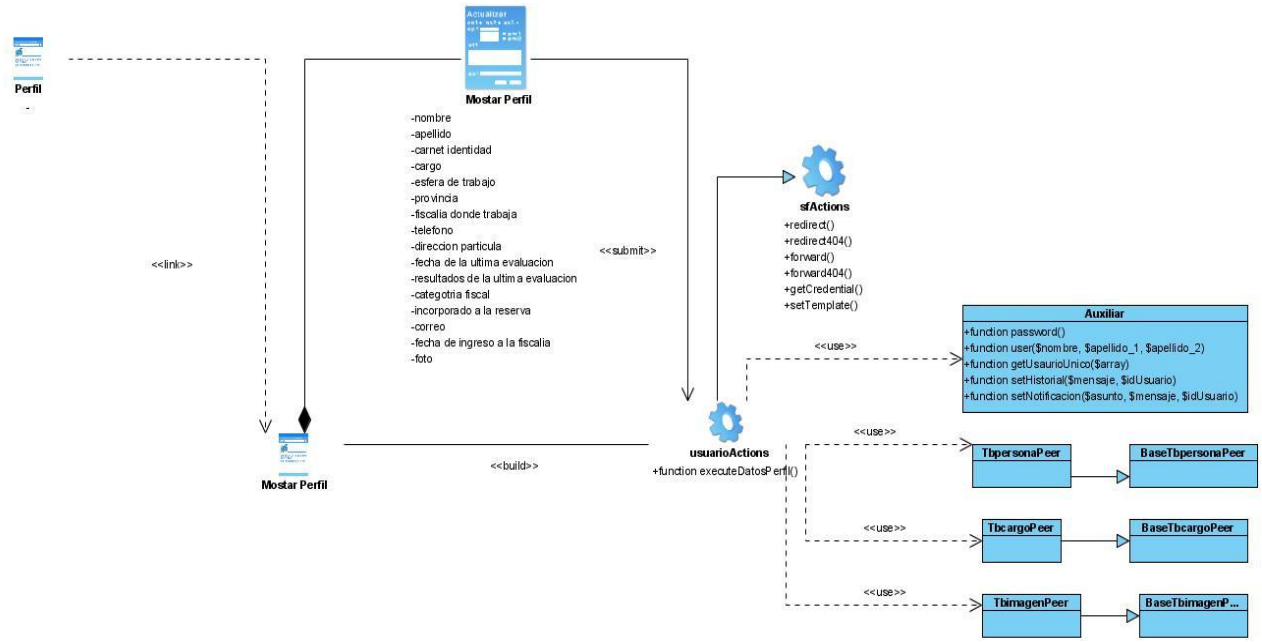

Figura 2. Diagramas de Clases: Mostrar Perfil.

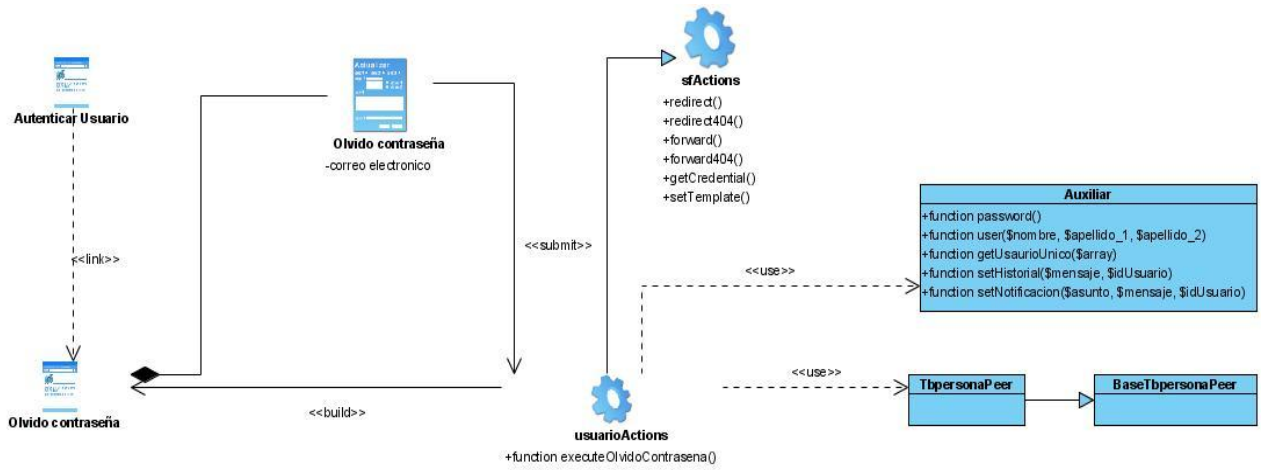

Figura 3. Diagramas de Clases: Olvidar Contraseña.

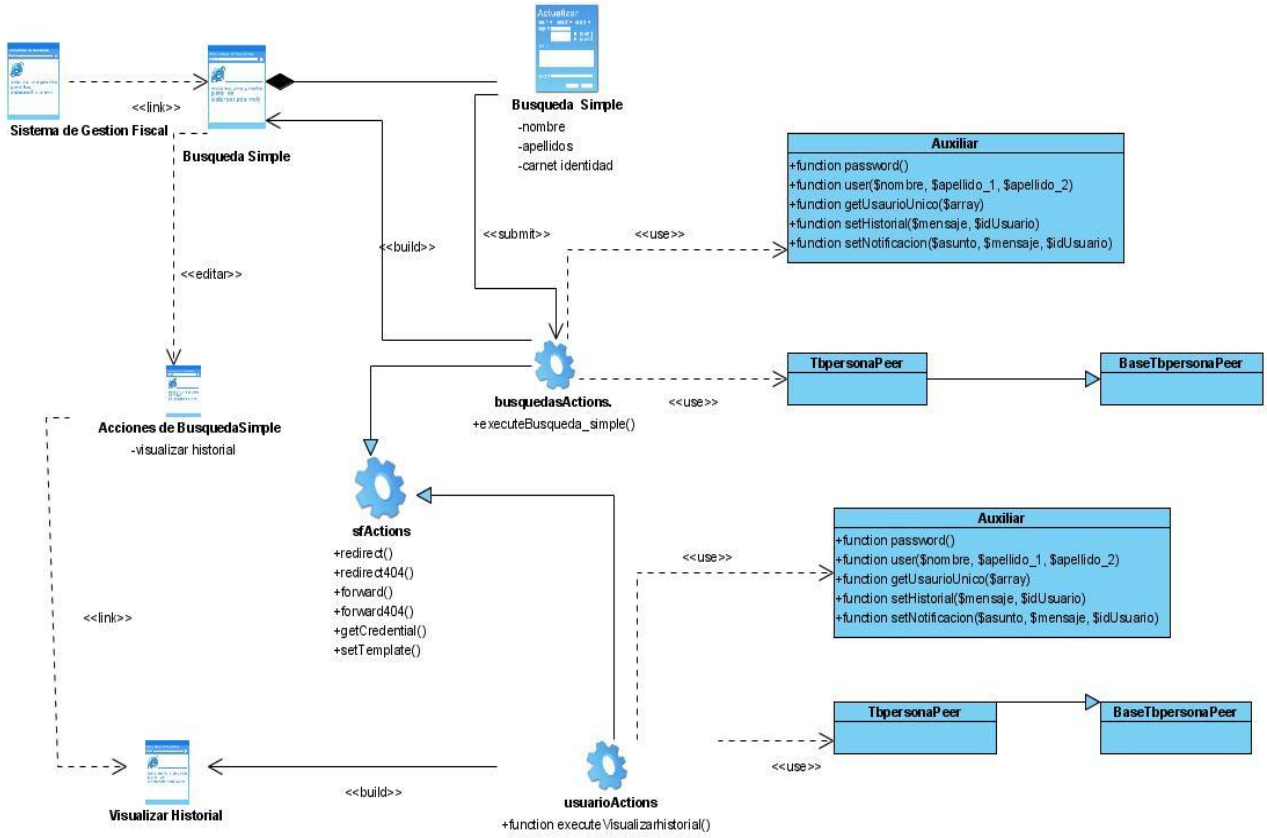

Figura 4. Diagramas de Clases: Visualizar Historial.

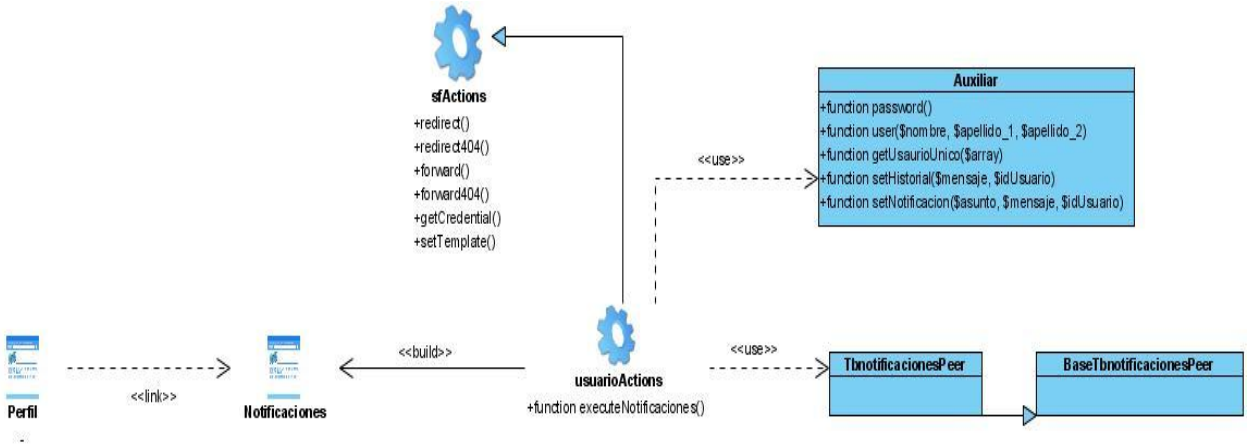

Figura 5. Diagramas de Clases: Visualizar Notificaciones.

Anexo 12. Diagramas de Clases del Diseño del Módulo Reporte y Búsqueda.

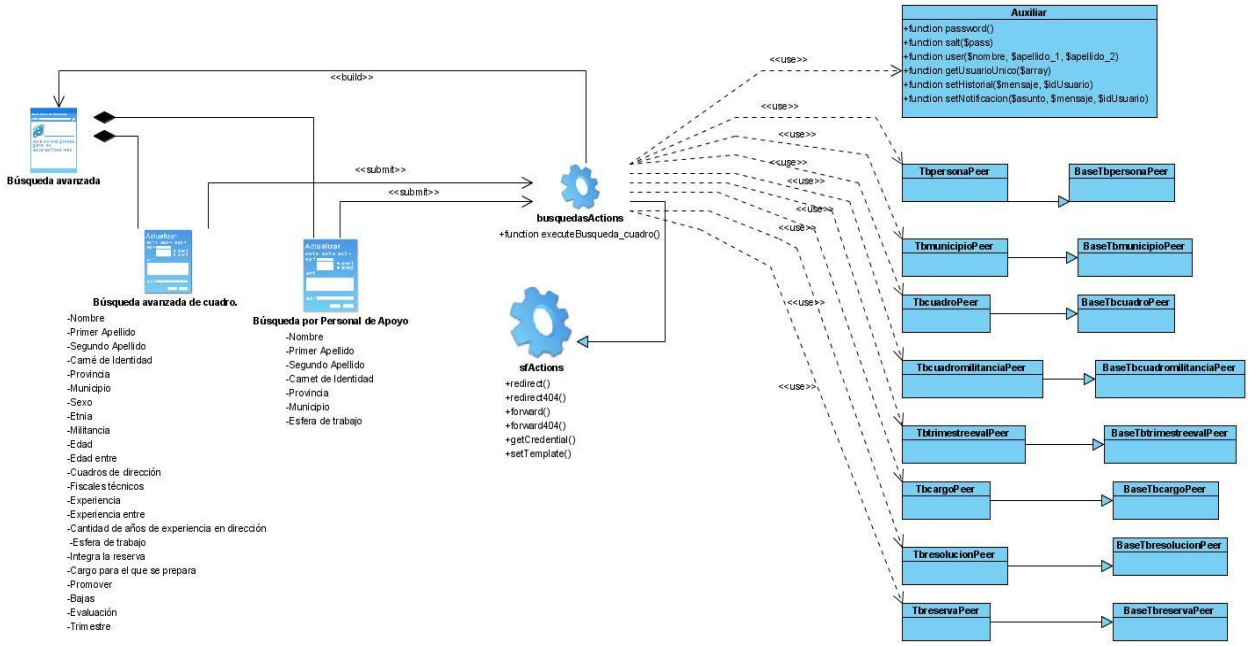

Figura 1. Diagramas de Clases: Búsqueda Avanzada de Cuadro.

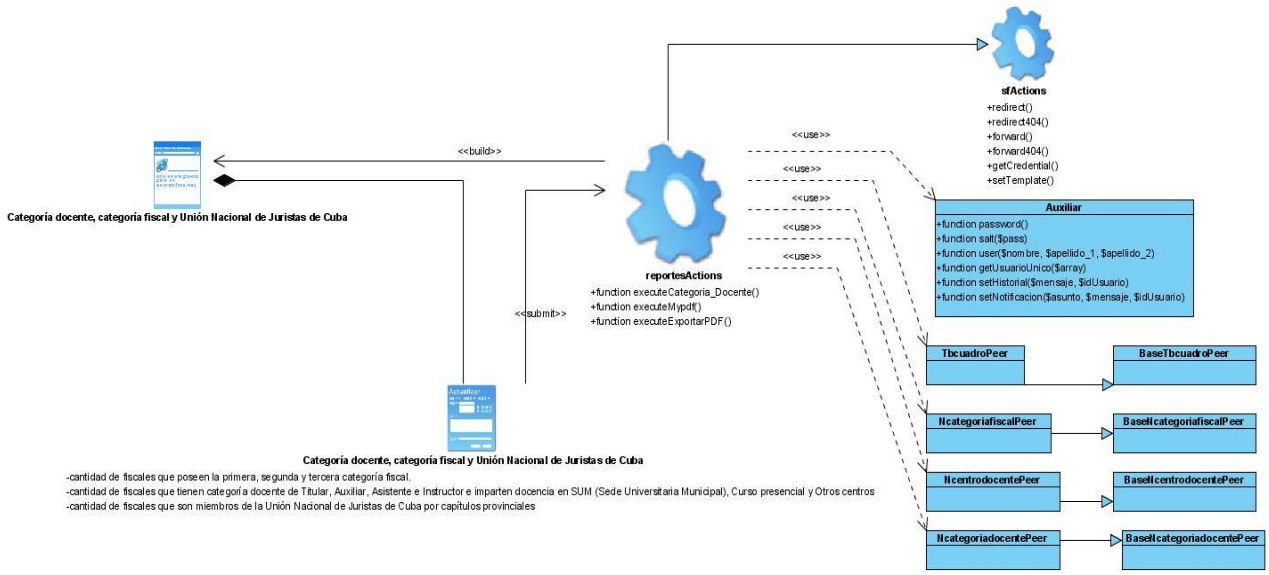

Figura 2. Diagramas de Clases: Categoría Docente, Categoría fiscal y Unión Nacional de Juristas de Cuba.

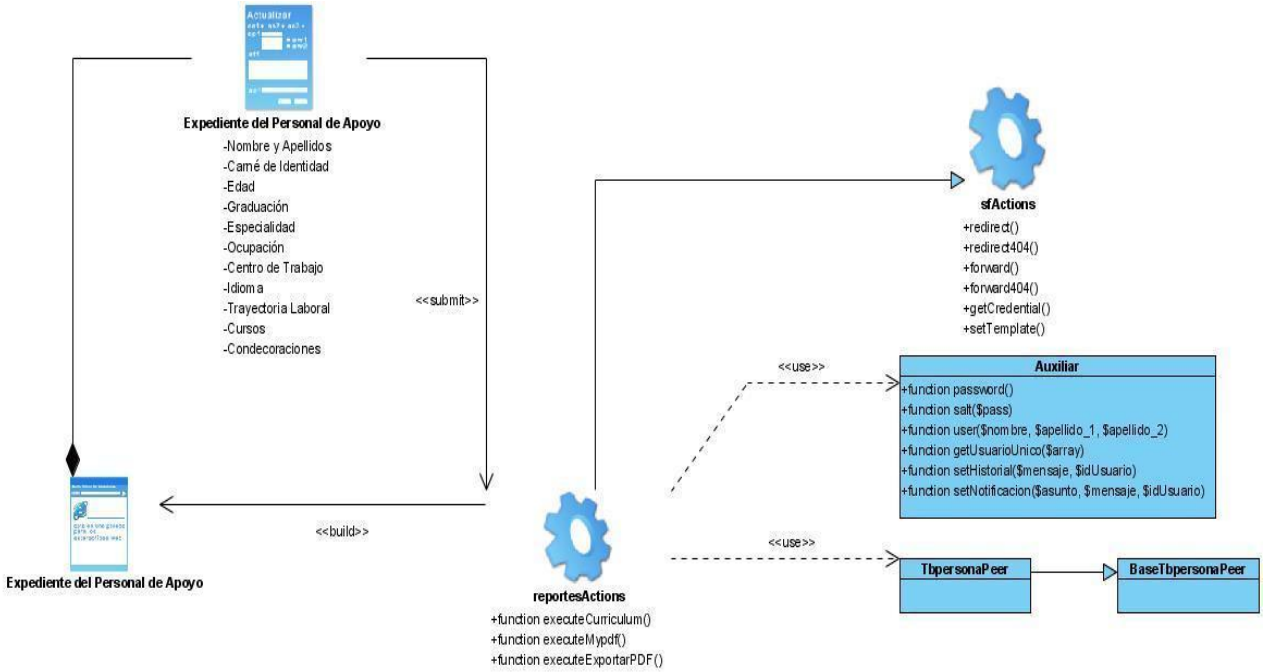

Figura 3. Diagramas de Clases: Generar Currículo del Cuadro.

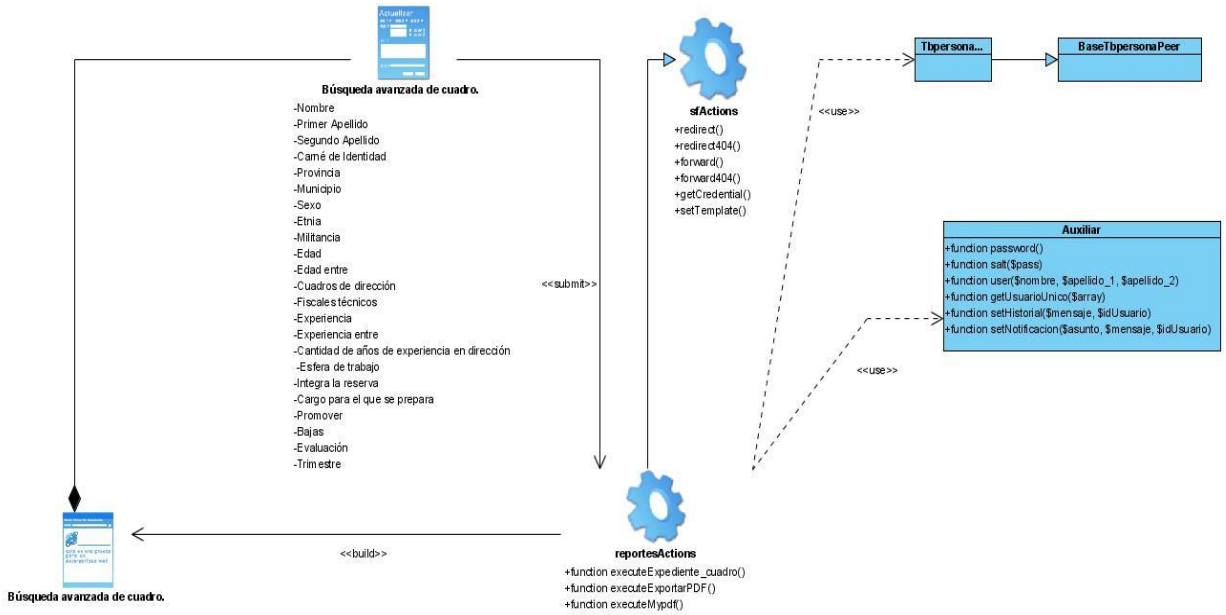

Figura 4. Diagramas de Clases: Generar Expediente del Cuadro.

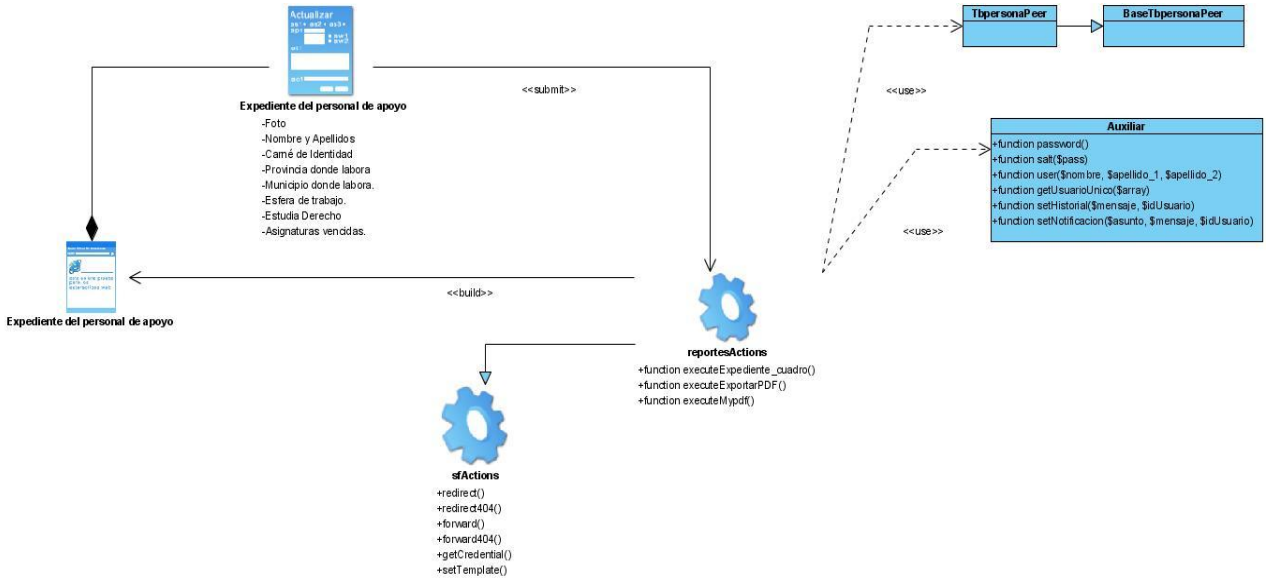

Figura 5. Diagramas de Clases: Generar Expediente del Personal de Apoyo.

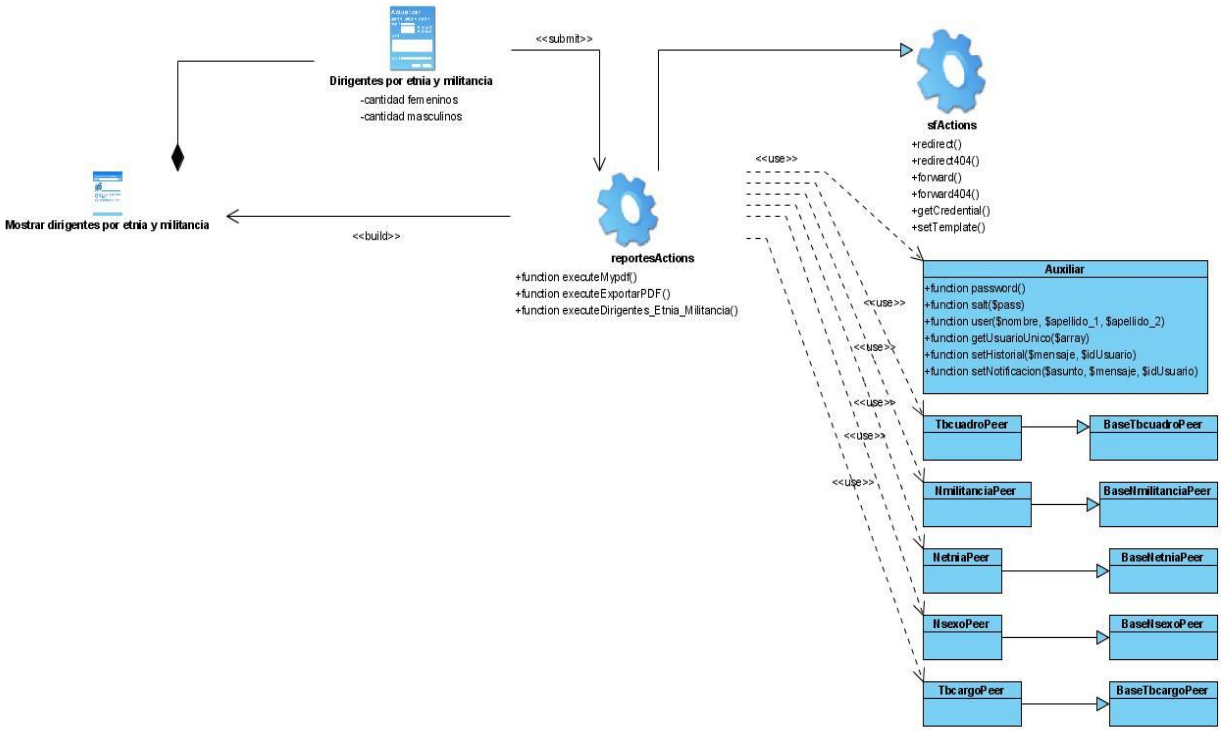

Figura 6. Diagramas de Clases: Mostrar Dirigentes por Etnia y Militancia.

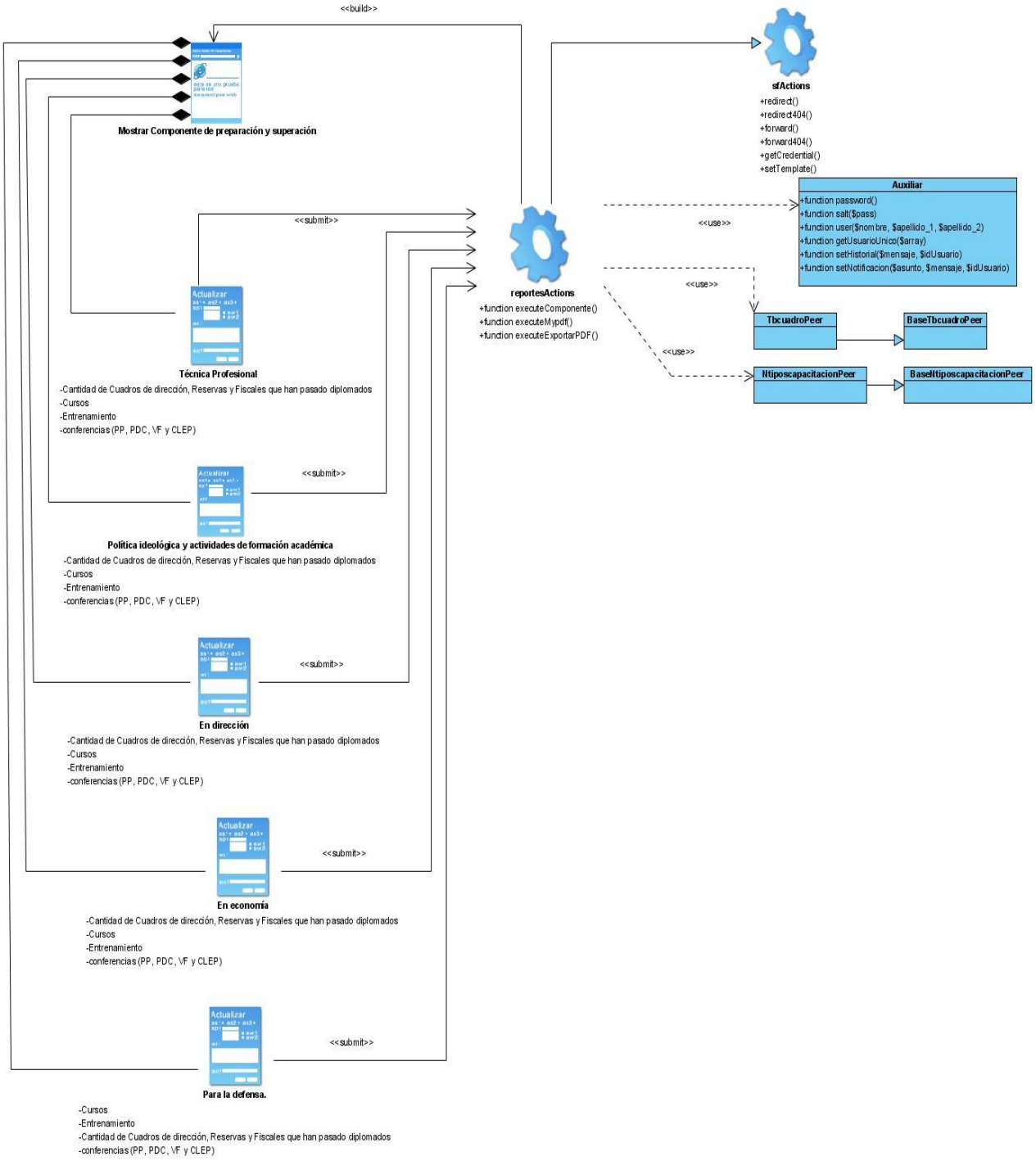

Figura 7. Diagramas de Clases: Mostrar Componente de Preparación y Superación.

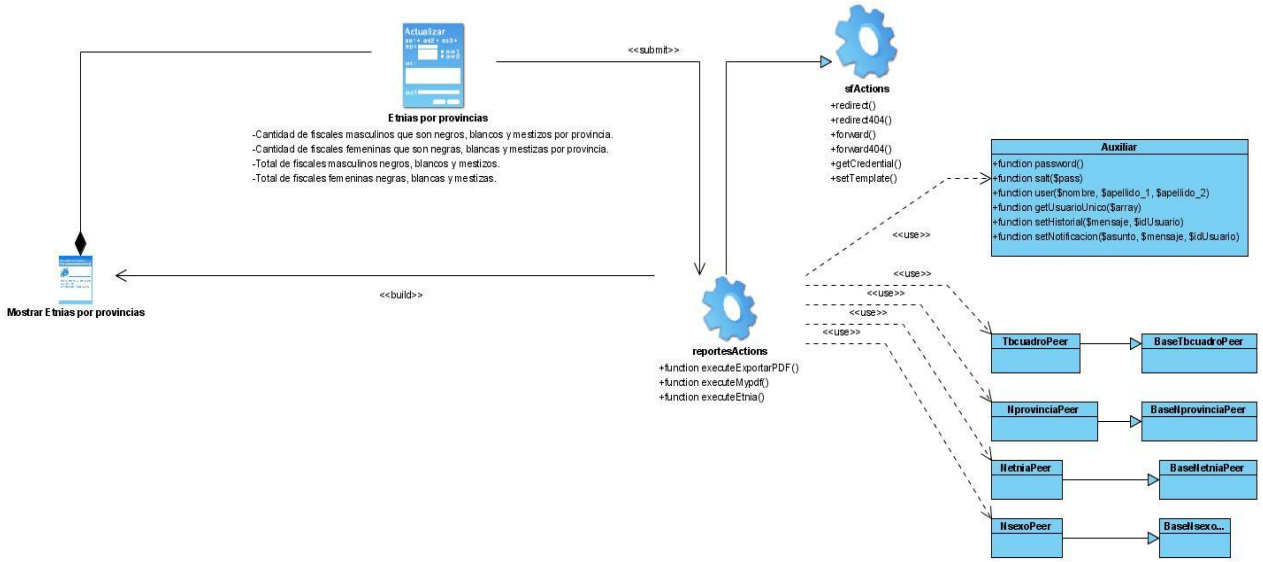

Figura 8. Diagramas de Clases: Mostrar Etnia por Provincias.

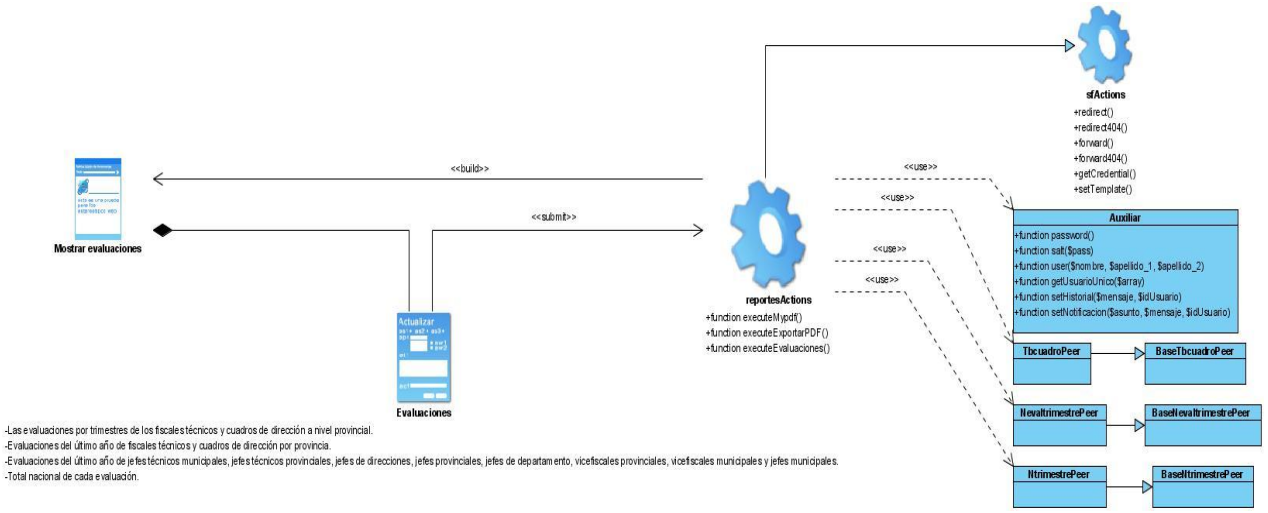

Figura 9. Diagramas de Clases: Mostrar Evaluaciones.

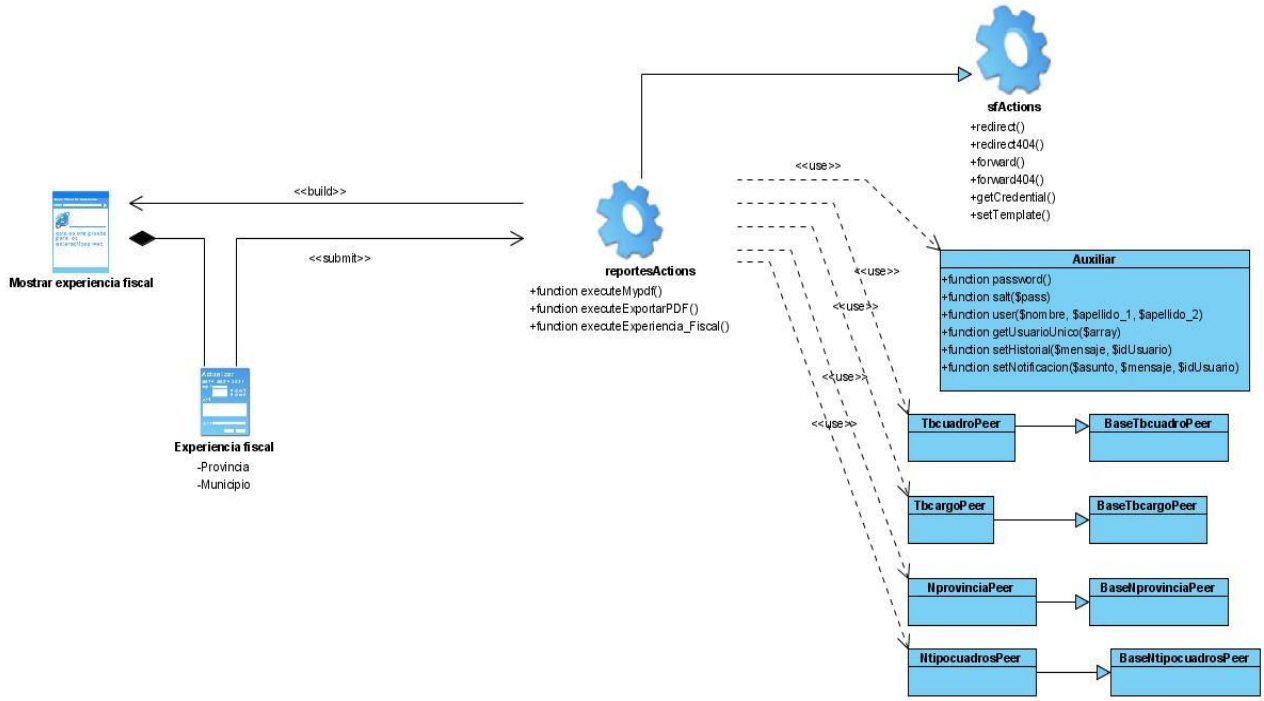

Figura 10. Diagramas de Clases: Mostrar Experiencia Fiscal.

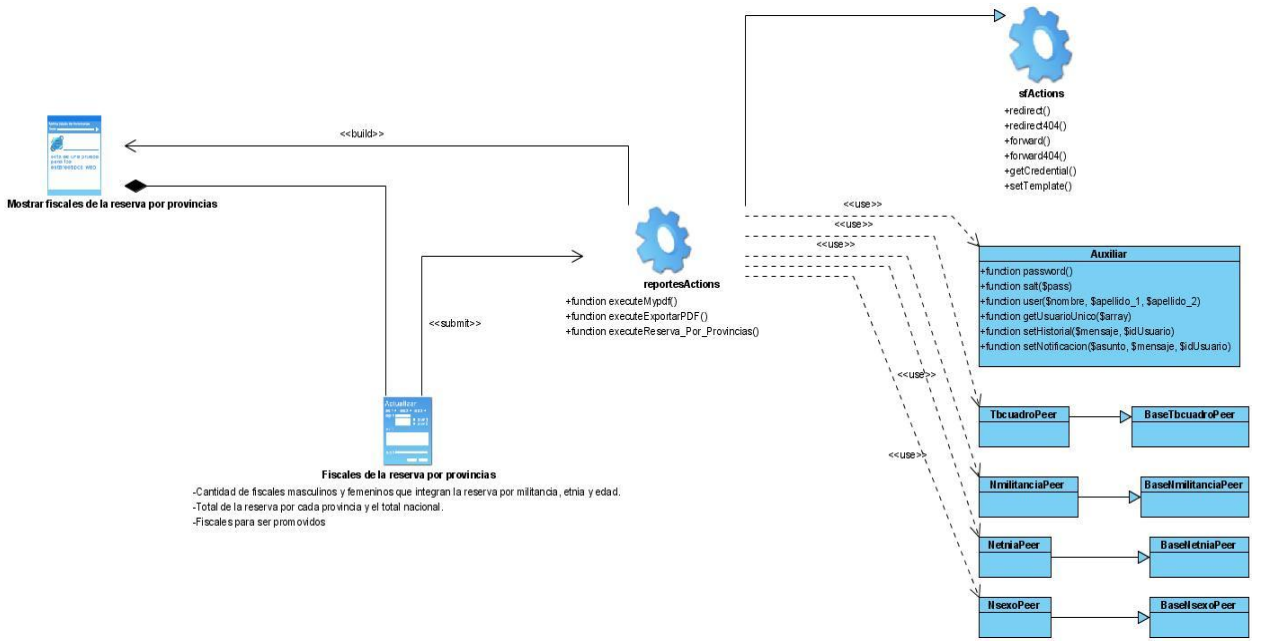

Figura 11. Diagramas de Clases: Mostrar Fiscales de la Reserva por Provincias.

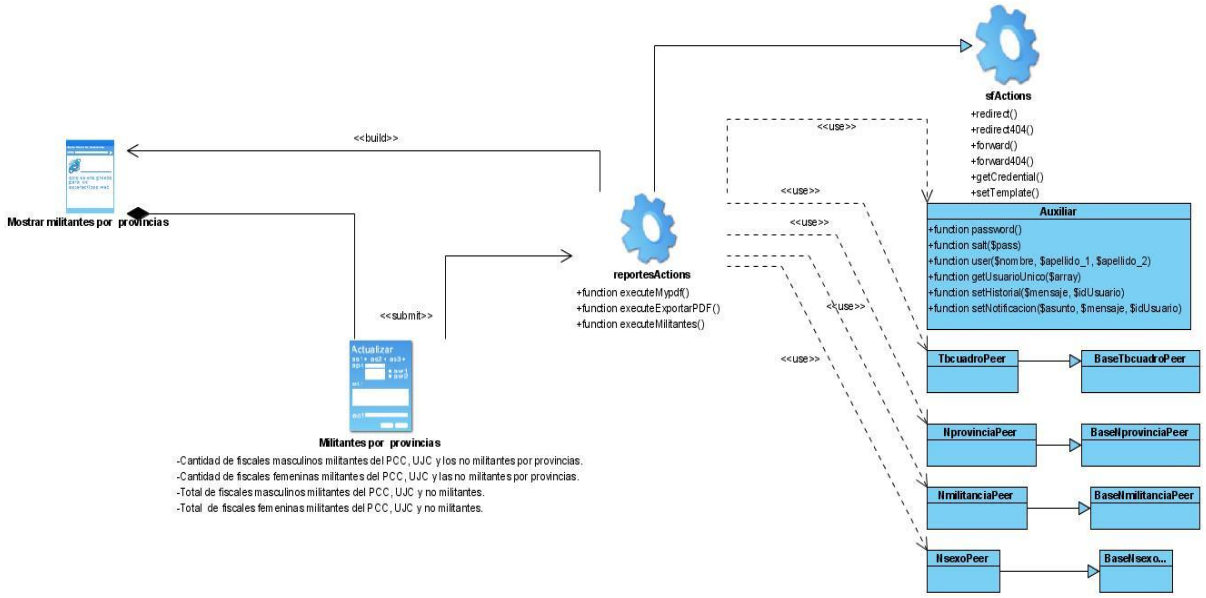

Figura 12. Diagramas de Clases: Mostrar Militantes por Provincias.

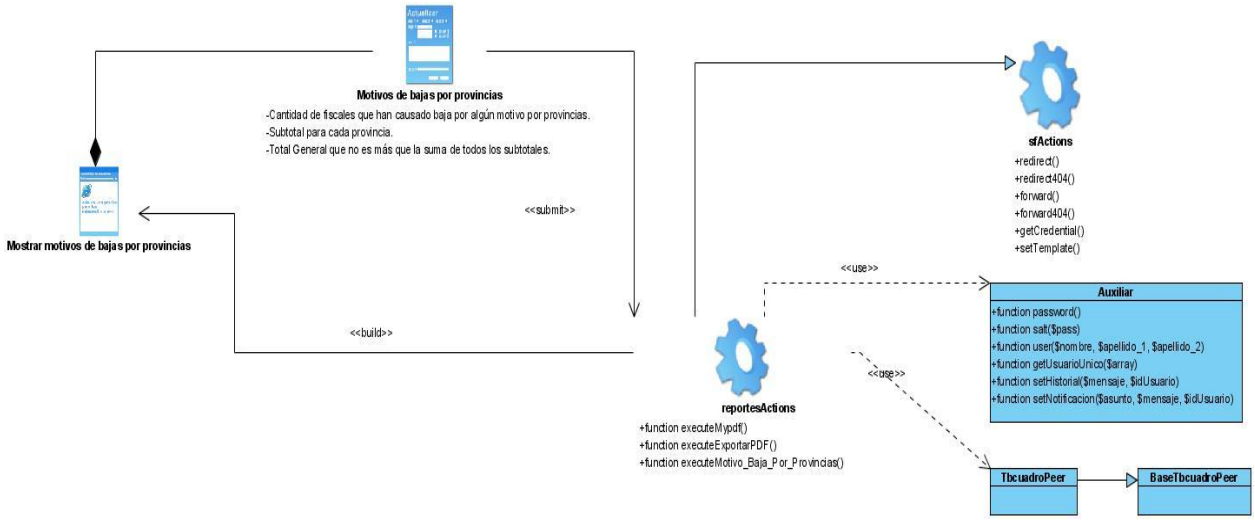

Figura 13. Diagramas de Clases: Mostrar Motivos de Bajas por Provincias.

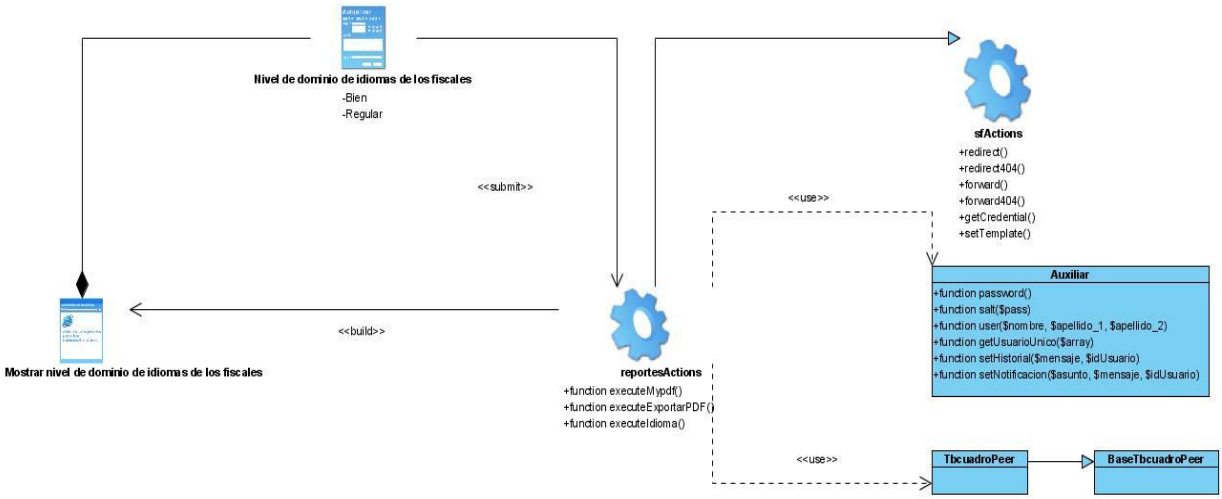

Figura 14. Diagramas de Clases: Mostrar Nivel de Dominio de Idiomas de los Fiscales.

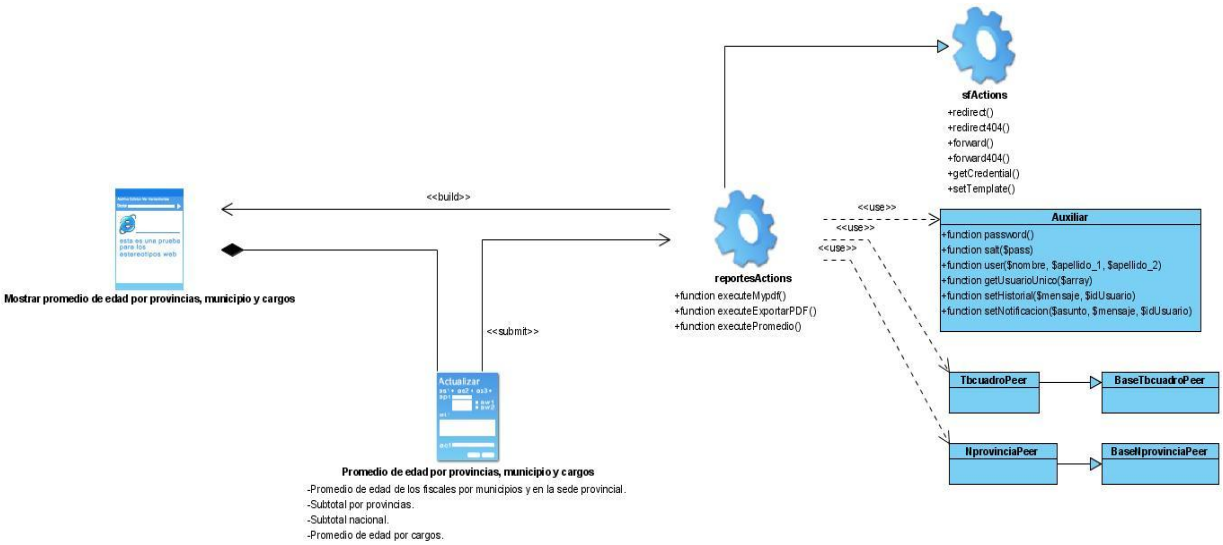

Figura 15. Diagramas de Clases: Mostrar Promedio de Edad por Provincias, Municipio y Cargos.

Anexo13. Descripción de las Clases de la Capa del Negocio.

Nombre:	busquedasActions
Tipo de clase:	controladora
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción**
executeBusqueda_cuadro()	Método encargado de realizar todas las acciones referentes la búsqueda de cuadro, registra los datos en la base de datos y muestra mensaje de notificación.
executeBusqueda_simple()	Método encargado de realizar la búsqueda simple y mostrar los resultados.
executeBusqueda_avanzada_capacitacion()	Método encargado de realizar la búsqueda avanzada de capacitación de los cuadros.
executeEditar_busqueda_simple()	Método encargado de editar la búsqueda simple.
Historial($mensaje)	Método encargado de registrar el historial cuando algún usuario realiza algún tipo de búsqueda.
executeIndex()	No realiza ninguna acción.

Nombre:	capacitacionCuadroActions
Tipo de clase:	controladora
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción**
executeIndex()	Método encargado de re direccionar para el método create, cuando carga la página.

executeCreate()	Se crea un nuevo módulo de capacitación de cuadro y cambia para la plantilla edit.
executeEdit()	Edita los datos de capacitación de cuadro.
executeVisualizar()	Visualiza los datos de capacitación.
executeUpdate()	Registra todos los datos de capacitación en la base de datos y registra el historial.
handleErrorUpdate()	Re direcciona para el método create en caso existir algún error.
validateUpdate()	Encargado de validar los datos.

Nombre:	captacionPlantillasActions
Tipo de clase:	controladora
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción**
executeIndex()	Método encargado de re direccionar para el método edit cuando carga la página.
executeEdit()	Edita los datos de captación de plantilla
executeUpdate()	Registra los datos de captación de plantilla en la base de datos y registra el historial.
handleErrorUpdate()	Re direcciona para el edit en caso de existir algún error.
validateUpdate()	Valida los campos.

Nombre:	captacionAsistenteFiscalActions
Tipo de clase:	Controladora
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción**
executeIndex()	Método encargado de re direccionar para el método create cuando carga la página.
executeCreate()	Se crea un nuevo módulo de captación de asistente fiscal y cambia para la plantilla edit.
executeEdit()	Edita los datos de captación de asistente fiscal.
executeVisualizar()	Visualizar los datos de captación de asistente fiscal.
executeUpdate()	Registra los datos de captación de asistente

	fiscal en la base de datos y registra el historial.
handleErrorUpdate()	Re direcciona para el create en caso de existir algún error.
validateUpdate()	Valida los campos.
executeAsignaturas()	Registra los datos de las asignaturas vencidas en la base de datos.
handleErrorAdicionarasignatura()	Re direcciona para la vista de asignatura en caso de existir algún error.
validateAdicionarasignatura()	Valida los campos de las asignaturas.

Nombre:	permisosActions
Tipo de clase:	Controladora
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción**
executeIndex()	Carga los permisos en la página que podrán ser asignados.
executeVisualizar()	Visualiza los permisos que puede tener el cuadro.
executeCargospermisos()	Registra los permisos asignados a la persona y guarda el historial.
executeVisualizarcargospermisos()	Visualiza los permisos asignados al cuadro.
Nombre:	captacionOtrasPersonasActions
Tipo de clase:	Controladora
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción**
executeIndex()	Método encargado de re direccionar para el método create cuando carga la página.
executeCreate()	Se crea un nuevo módulo de captación de otras personas y cambia para la plantilla edit.
executeEdit()	Edita los datos de captación de otras personas.
executeVisualizar()	Visualizar los datos de captación de otras personas.
executeUpdate()	Registra los datos de captación de otras personas en la base de datos y registra el historial.
handleErrorUpdate()	Re direcciona para el create en caso de existir algún error.
validateUpdate()	Valida los campos.

Nombre:	categoriaDocenteFiscalActions
Tipo de clase:	Controladora
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción**
executeIndex()	Re direcciona para el método Create cuando carga la página.
executeCreate()	Crea un una nueva categoría docente fiscal y re direcciona para la plantilla edit.
executeEdit()	Permite editar los datos de la categoría docente fiscal.
executeUpdate()	Registra los datos de la categoría docente fiscal y registra el historial.
handleErrorUpdate()	Re direcciona para el método Create en caso de existir algún error.
validateUpdate()	Valida los datos.
executeVisualizar()	Visualiza los datos de la categoría docente fiscal.
Nombre:	reportesActions
Tipo de clase:	Controladora
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción**
executeIndex()	En este método se obtienen los diferentes tipos de superaciones que se utilizan en los reportes.
executeExportarPDF()	Método para exportar el reporte en pdf.
executeMypdf()	Convierte la el reporte en la página html en formato pdf.
executeIdioma()	Reporte que muestra los cuadros que dominan Bien o Regular un Idioma
executeEtnia()	Reporte que muestra la etnia por provincias
executeMilitantes()	Reporte que muestra los militantes por provincias
executePromedio()	Reporte que muestra promedio de edad por provincias, municipios y cargos
executeDatos_del_perfil()	Reporte que muestra promedio de edad por provincias, municipios y cargos
executeDirigentes_Etnia_Militancia()	Reporte que muestra en dependencia de los cargos, datos sobre la etnia y militancia del los dirigentes masculinos.
executeReserva_Por_Provincias()	Reporte que muestra las reservas por provincia y las reservas para promover.
executeEvaluaciones()	Reporte que muestra evaluaciones por

	trimestres, tipo cuadros y cargos.
executeMotivo_Baja_Por_Provincias()	Reporte que muestra los motivos de bajas por provincia.
executeCategoria_Docente()	Reporte que muestra datos sobre la Categoría Docente y fiscal
executeAsistentes_Fiscales()	Reporte que muestra datos sobre los Asistentes Fiscales
executeExperiencia_Fiscal()	Reporte que muestra la Experiencia Fiscal por provincias y municipios
executePlantilla_Fiscalía()	Reporte que muestra la Plantilla Fiscal por provincias y municipios
executeComponente()	Reporte que muestra el componente de preparación y superación técnica profesional
executeExpediente_cuadro()	Reporte que muestra los expedientes de cuadro.
executeExpediente_asistente()	Reporte que muestra los expedientes de asistente fiscal.
executeExpediente_otras_personas()	Reporte que muestra los expedientes de otras personas.
executeCurriculum()	Reporte que muestra el currículo.
Historial($mensaje)	Método que registra el historial.
executeResolucion()	Reporte que muestra las resoluciones.

Nombre:	resolucionActions
Tipo de clase:	Controladora
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción**
executeVinculos()	Direcciona para la plantilla vínculo para crear una nueva resolución.
executeCreatedesignacion()	Crea una nueva designación y direcciona para el método executeDesignacion para terminar de crearla.
executeCreatemovimiento()	Crea un movimiento y direcciona para el método executeMovimiento para terminar de crearlo.
executeEdit()	Edita las resoluciones.
executeDesignacion()	Crea una designación para ser insertada.
executeMovimiento()	Crea un movimiento para ser insertado.
executeInsertarmovimiento()	Registra los datos del movimiento creado en la base de datos y guarda el historial.
executeInsertardesignacion()	Registra la designación creada en la base de datos y guarda el historial.
executeVisualizardesignacion()	Visualiza los datos de las designaciones.

executeVisualizarmovimiento()	Visualiza los datos de los movimientos.
handleErrorInsertardesignacion()	Re direcciona para designación en caso de existir algún error.
validateInsertardesignacion()	Valida los datos.
handleErrorInsertarmovimiento()	Re direcciona para movimiento en caso de existir un error.
validateInsertarmovimiento()	Valida los datos.

Nombre:	resolucionCeseActions
Tipo de clase:	Controladora
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción**
executeCreate()	Crea un nuevo cese de funciones y direcciona para la vista de editar. edit
executeEdit()	Permite editar los datos del cese de funciones.
executeUpdate()	Permite registrar los datos del cese en la base de datos y registra el historial y re direcciona para 'cesefuncion'. Para mostrar los datos del mensaje a realizar.
executeVisualizarcese()	Permite visualizar los datos del cese de funciones.
handleErrorUpdate()	Re direcciona para 'cesefuncion' en caso de existir algún error
validateUpdate()	Valida los datos.
executeCesefuncion()	Muestra en pantalla los datos registrados y el mensaje de la resolución (en esta vista pueden ser cambiados).

Nombre:	acuerdoActions
Tipo de clase:	Controladora
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción**
executeIndex()	Cuando carga la página re direcciona para el método list.
executeList()	Realiza una consulta directa a la base de datos.
executeShow()	Muestra un acuerdo.
executeCreate()	Crea un nuevo acuerdo y re direcciona para el edit.
executeEdit()	Permite editar los datos de los acuerdos.
executeAsambleanacional()	Carga en la página los datos del acuerdo de la asamblea nacional.
executeUpdate()	Registra los datos del acuerdo de la asamblea nacional y guarda el historial.
executeConsejoestado()	Carga en la página los datos del acuerdo del consejo de estado.
executeUpdateCE()	Registra los datos del acuerdo del consejo de estado y guarda el historial.
executeDelete()	Permite eliminar un acuerdo.
handleErrorUpdate()	Re direcciona para Asambleanacional en caso de existir algún error.
validateUpdate()	Valida los datos de la asamblea nacional.
executeVisualizarasambleanacional()	Visualiza los datos del acuerdo de la asamblea nacional.
handleErrorUpdateCE()	Re direcciona para el método Consejoestado en caso de existir algún error.
validateUpdateCE()	Valida los datos del acuerdo del consejo de estado.
executeVisualizarconsejoestado()	Visualiza los datos del acuerdo del consejo de estado.

Nombre:	acuerdoCeseActions
Tipo de clase:	Controladora
Atributo	**Tipo**

Para cada responsabilidad:	
Nombre:	**Descripción**
executeIndex()	Cuando carga la página re direcciona para el método list.
executeList()	Realiza una consulta directa a la base de datos.
executeShow()	Muestra un acuerdo.
executeCreate()	Crea un acuerdo para el cese y re direcciona para el edit.
executeEdit()	Permite editar un acuerdo para el cese.
executeConsejoestado()	Carga en la página los datos del acuerdo del consejo de estado para el cese.
executeUpdateCE()	Registra los datos del acuerdo del consejo de estado para el cese, y guarda el historial.
executeAsambleanacional()	Carga en la página los datos del acuerdo de la asamblea nacional para el cese.
executeUpdateAN()	Registra los datos del acuerdo de la asamblea nacional para el cese y guarda el historial.
executeDelete()	Permite eliminar un acuerdo para el cese.
executeVisualizarasambleanacional()	Visualiza los datos del acuerdo de la asamblea nacional para el cese.
executeVisualizarconsejoestado()	Visualiza los datos del acuerdo del consejo de estado para el cese.
handleErrorUpdateCE()	Re direcciona para 'consejoestado' en caso de existir algún error.
validateUpdateCE()	Valida los datos del acuerdo para el cese del consejo de estado.
handleErrorUpdateAN()	Re direcciona para 'asambleanacional' en caso de existir algún error.
validateUpdateAN()	Valida los datos del acuerdo de la asamblea nacional para el cese.

Nombre:	cuadroActions
Tipo de clase:	Controladora
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción**
executeIndex()	Cuando carga la página re direcciona para el método Create.
executeList()	Realiza una consulta directa a la base de datos.

executeShow()	Muestra los datos de una persona.
executeCreate()	Crea un cuadro y re direcciona para el edit.
executeEdit()	Permite editar los datos del cuadro.
executeUpdate()	Registra los datos del cuadro y registra el historial.
executeDelete()	Permite eliminar una persona.
executeReservas()	Carga los datos de la reserva para ser registrados.
executeUpdatereservas()	Registra los datos de la reserva y guarda el historial.
executeLaborales()	Carga los datos laborales para ser insertados.
executeUpdatelaborales()	Registra los datos laborales y registra el historial.
executeAdicionarviajes()	Permite registrar los datos de los viajes.
executeIdiomas()	Permite registrar los datos del idioma.
executeDocumentos()	Permite registrar los datos de los documentos.
handleErrorUpdate()	Re direcciona par el método create cuando existe algún error.
validateUpdate()	Valida los datos.
executeSancion()	Permite registrar las datos de la sanciones.
executeVisualizarpersonales()	Visualiza los datos personales.
executeVisualizarlaborales()	Visualiza los datos laborales.
executeVisualizarreservas()	Visualiza los datos de la reserva.

Nombre:	Auxiliar	
Tipo de clase:	Secundaria.	
Atributo		**Tipo**
Para cada responsabilidad:		
Nombre:		**Descripción**
password()		Contraseña aleatoria
salt($pass)		En cripta la contraseña a md5
user($nombre, $apellido_1, $apellido_2)		Convierte todo a minúscula, Quita los espacios, tildes, del nombre, devuelve el usuario.
getUsuarioUnico($array)		Devuelve un usuario único.
setHistorial($mensaje, $idUsuario)		Registra el Historial.
setNotificacion ($asunto,$mensaje,$idUsuario)		Registra las notificaciones.

Nombre:	myUser
Tipo de clase:	Secundaria.
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción**
getUsuarioId()	Retorna el usuario según el id.
getNombreApellido()	Retorna el nombre y apellidos.
getCargo()	Retorna el cargo.
getEmail()	Retorna el correo.
signIn($user)	Según el id del usuario adiciona las credenciales de trabajo y del usuario.
signOut()	Según el usuario limpia las credenciales de trabajo y del usuario.

GLOSARIO DE TERMINOS

GNU/Linux: GNU/Linux es, un Sistema Operativo. Es una implementación de libre distribución UNIX para computadoras personales (PC), servidores, y estaciones de trabajo. Fue desarrollado para el i386 y ahora soporta los procesadores i486, Pentium, Pentium Pro y Pentium II y superior, así como los clones AMD y Cyrix. También soporta máquinas basadas en SPARC, DEC Alpha, PowerPC/PowerMac, y Mac/Amiga Motorola 680x0.

Framework: Denota la infraestructura sobre la cual se reúnen un conjunto de lenguajes, herramientas y servicios que simplifican el desarrollo de aplicaciones en entornos de ejecución distribuido.

UML: Lenguaje Unificado de Modelado, es un lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software.